

<http://www.ksi.mff.cuni.cz/~svoboda/courses/222-NSWI170/>

Cvičení

NSWI170: Počítačové systémy

2022/23 LS

Martin Svoboda

martin.svoboda@matfyz.cuni.cz

Univerzita Karlova, Matematicko-fyzikální fakulta

Cvičení 1: **Základy jazyků C a C++**

Používané nástroje

Mattermost

- <https://ulita.ms.mff.cuni.cz/mattermost/>
 - [.../ar2223ls/channels/nswi170-compsys-svoboda](https://ulita.ms.mff.cuni.cz/ar2223ls/channels/nswi170-compsys-svoboda)

ReCodEx

- <https://recodex.mff.cuni.cz/>

Coliru

- <http://coliru.stacked-crooked.com/>

Arduino IDE

- <https://www.arduino.cc/en/software/>

P11: Hello World

Naprogramujte jednoduchou **Hello World** aplikaci

- Aneb na standardní výstup vypíšeme pozdrav *Hello World*
- Menší nápověda
 - `#include <stdio.h>`
 - `int main(int argc, char** argv) { ... }`
 - `int main() { ... }`
 - `printf("...");`
 - `\n`

P12: Vánoční stromeček

Vypište na standardní výstup **vánoční stromeček**

- Velikost stromečku je určena jeho výškou
 - V každé úrovni vypíšeme příslušný počet **hvězdiček**
 - Ty navíc zleva zarovnáme pomocí mezer
- Ukázka výstupu pro stromeček velikosti 4

```
 *
 ***
 *****
 ********
```

- Kód vhodně dekomponujeme do jednotlivých funkcí
- Opět menší nápověda
 - `printf("%c", '...');`
 - `for (int i = 0; i < height; ++i) { ... }`

P13: Celočíselný průměr

Spočítejte **celočíselný průměr** zadaných přirozených čísel

- Vstup předpokládejme v podobě lokální proměnné
 - `int numbers[] = { 6, 10, 12, 8 };`
- Nejprve spočítáme příslušný průměr
- Vypíšeme jej pomocí odpovídajícího počtu **hvězdiček**
 - Např. `*****`
- Náповěda
 - `int average(int numbers[], int count) { ... }`
 - `int size = sizeof(numbers) / sizeof(numbers[0]);`

P14: Klouzavý průměr

Spočítejte **klouzavé průměry** zadaných přirozených čísel

- Vstup předpokládejme v podobě konstantního výrazu
 - `constexpr int numbers[] = { 3, 8, 5, 7, 2, 5 };`
- Ukázka očekávaného výstupu
 - Pro výše uvedená vstupní čísla a okénko velikosti 3

```
*****  
*****  
****  
****
```

- Doporučené rozhraní
 - `void averages(
 const int numbers[], int count, int window
) { ... }`

Cvičení 2: **Arduino: Diody**

Arduino

Platforma Arduino

- **Arduino UNO**
 - Základní deska, 14 digitálních a 6 analogových pinů
 - CPU ATmega328P, 16 MHz, FLASH memory 32 kB
- **Funduino**
 - Multifunction shield
 - Diody, tlačítka, segmentový displej, ...
- Dokumentace
 - <https://docs.arduino.cc/>
 - <http://kabinet.fyzika.net/dilna/ARDUINO/funduino-popis.php>

Arduino

Arduino IDE

- Základní ovládání
 - **CTRL+S**: uložení souboru (přípona *.ino)
 - **CTRL+R**: kompilace programu
 - **CTRL+U**: nahrání programu do Arduina

Struktura programu

- Funkce `void setup()` ;
 - Spustí se jednou při startu
 - Obsahuje nejrůznější inicializace
 - Např. nastavení módů pinů, počátečních hodnot, ...
- Funkce `void loop()` ;
 - Obsahuje vlastní výkonný kód
 - Spouští se neustále dokola, přibližně 1000x za sekundu

Arduino: Diody

Diody D1 až D4

- Dostupné přes piny 13 až 10
 - **Konstanty pro čísla pinů** `led1_pin`, `led2_pin`, ..., `led4_pin`
- Pro označení diod budeme používat **logická čísla 0 až 3**
 - Kvůli dosažení vyšší úrovně abstrakce
- **Inicializace programu**
 - Nastavení módů pinů
 - `void pinMode(pin, OUTPUT);`
 - Explicitní vypnutí všech diod
- **Ovládání diody**
 - Zápis `LOW` (zapnutí) / `HIGH` (vypnutí) na příslušný pin
 - `void digitalWrite(pin, value);`

P21: Rozsvícení diody

Rozsviďte konkrétní vybranou diodu

- Nejprve připojíme soubor s definicemi potřebných konstant
 - `#include "funshield.h"`
 - <https://www.ksi.mff.cuni.cz/teaching/nswi170-web/downloads/Funshield.zip>
 - Soubor `funshield.h` umístíme do adresáře našeho programu
- **Překladové pole** z čísel diod na čísla pinů
 - `constexpr int diodePins[] =
 { led1_pin, led2_pin, led3_pin, led4_pin };`
- Potřebný kód zapouzdříme do následujících funkcí
 - `void diodeInitialize(int number);`
 - `void diodeChange(int number, bool state);`
- Nakonec vše otestujeme zapnutím jedné konkrétní diody

P22: Rozblikání diody

Rozblikujte konkrétní vybranou diodu

- **Kontrola časování**
 - `unsigned long currentTime = millis();`
 - Vrátil aktuální **systemový čas** v milisekundách
 - Základní myšlenka detekce okamžiku další události
 - `if (currentTime >= previousTime + periodLength)`
`{ ... }`
 - Ve skutečnosti ale musíme hlídat i **přetečení hodnoty času**
 - To nastane přibližně po 50 dnech
 - Trik pro zjištění maximální hodnoty: `~(unsigned long)0`
- Potřebné údaje budeme ukládat do **globálních proměnných**
 - Čas předchozí události, logický stav zapnutí diody
- Kód otestujeme rozblikáním jedné vybrané diody
 - Délka intervalu jako pojmenovaná konstanta, např. 500 ms

P23: Železniční semafor

Rozblikujte železniční semafor

- Aneb střídavě budeme rozsvěcovat dvojice sousedních diod
- Současně **refaktorujeme** stávající kód pracující s diodami
 - Jednotlivé **diody začneme reprezentovat pomocí objektů**
 - Tedy instancí vhodné navržené třídy
 - Ta bude obsahovat nejenom potřebné datové položky, ...
 - ... ale také zapouzdří požadovanou funkcionalitu
 - **Instance diod budeme držet v globálním poli**
 - `Diode diodes[diodesCount];`
 - Inicializace pak provedeme v rámci funkce `setup()`
- Analogickým způsobem **navrhne i třídu pro časovač**

P23: Železniční semafor

Kostra třídy pro reprezentaci diody

```
class Diode {
private:
    int diodeNumber_;
    bool currentState_;
public:
    void initialize(int diodeNumber) {
        ...
    }
    void change(bool newState) {
        ...
    }
    void change() {
        ...
    }
};
```

Cvičení 3: **Arduino: Tlačítka**

Arduino: Tlačítka

Tlačítka B1 až B3

- Piny `button1_pin`, `button2_pin` a `button3_pin`
 - Opět ale chceme pracovat na vyšší úrovni abstrakce
 - Aneb používat pro tlačítka **logická čísla 0 až 2**
- **Inicializace tlačítek**
 - `void pinMode(pin, INPUT);`
- **Detekce stisknutí**
 - Čtení `LOW` (stisknuto) / `HIGH` (uvolněno) na příslušném pinu
 - `int digitalRead(pin);`

P31: Stisknutí tlačítka

Signalizujte stav stisknutí tlačítka rozsvícením příslušné diody

- **Překladové pole** pro čísla pinů tlačítek
 - `constexpr int buttonPins[] =
 { button1_pin, button2_pin, button3_pin };`
- Veškerou funkcionalitu zapouzdříme do vlastní třídy
 - A to podobně jako u diod

P32: Ovládání diod

Stisknutím tlačítka změňte stav příslušné diody

- Aneb danou diodu zapneme / vypneme
 - Nezáleží na tom, po jak dlouhou dobu bude tlačítko stisknuto

P33: Odskakování tlačítka

Vyřešte problém špatné detekce stisknutí tlačítka

- Ten je způsoben mechanickými vlastnostmi tlačítek
 - Konkrétně naše tlačítka navíc nemají ani příliš velkou kvalitu
 - Mohou tedy sama od sebe generovat **krátké zákmity**
- Velmi krátké změny stavu proto jednoduše **odfiltrujeme**
 - Aneb budeme je ignorovat
 - Předpokládejme konkrétně interval např. 10 ms
- Současně stávající kód refaktorujeme
 - Nově **vyčleníme funkci tlačítka na detekci vzniku události stisknutí / uvolnění a oddělíme ji od dotazů na tyto události**
 - Díky tomu bude možné takové dotazy volat i opakovaně
 - Tedy opakovaně během jednoho volání hlavní smyčky `loop`

P34: Binární rozklad

Pomocí diod zobrazujte hodnotu inkrementovaného počítadla

- Počítadlo začíná na 0 a inkrementuje se o 1 každou 1 sekundu
 - Povolené hodnoty počítadla jsou jen v intervalu 0 až 15
 - Při přetečení se vrátíme zpět na 0
- Pomocí diod vždy **zobrazíme nejnižší 4 bity aktuálního čísla**
 - Bit 1 diodu rozsvítí, bit 0 naopak zhasne
 - Např. pro číslo $5_{10} = 101_2$ zobrazíme 0101
 - Aneb dioda číslo 0 nesvítí, 1 ano, 2 ne, 3 ano
 - Menší nápověda
 - Bitová konjunkce $x \& 1$, bitový posun $x \ll 1$
- Veškerou funkcionalitu počítadla zapouzdříme do vlastní třídy
- Stisknutí tlačítka B1 způsobí resetování aktuální hodnoty na 0

Cvičení 4: **Arduino: Displej I**

Arduino: Sériová linka

Sériová linka

- **Inicializace** obousměrného spojení
 - Náš program: funkce `setup`
 - `Serial.begin(9600);`
 - Arduino IDE: Tools / Serial monitor
 - Nastavit stejnou rychlost
- **Posílání textu**
 - Funkce `Serial.print(...)` nebo `println(...)`
 - Různé varianty pro čísla, symboly nebo celé řetězce

P41: Jednoduchý časovač

Pomocí sériové linky vypisujte aktuální čas

- Aneb jeho hodnotu pravidelně posílejte z Arduina do počítače
 - A to každou sekundu
 - Hodnotu ořízneme na celé sekundy

Arduino: Displej

Segmentový displej

- **Piny** `latch_pin`, `data_pin` a `clock_pin`
 - Inicializujeme je v módu `OUTPUT`
- Postup **zobrazení konkrétního glyfu**
 - Zavřeme zámek
 - `digitalWrite(latch_pin, LOW);`
 - Pošleme masku glyfu
 - `shiftOut(data_pin, clock_pin, MSBFIRST, glyphMask);`
 - Pošleme masku pozice
 - `shiftOut(data_pin, clock_pin, MSBFIRST, positionMask);`
 - Otevřeme zámek
 - `digitalWrite(latch_pin, HIGH);`

Arduino: Displej

Segmentový displej (pokračování)

- **Reprezentace glyfu**

- `byte glyphMask = 0bHGFEDCBA;`
 - Popíšeme stav každého jednotlivého segmentu
 - Bit 0 (svítí), bit 1 (nesvítí)
 - Mapování segmentů: od horního (A) ve směru hodinových ručiček, následně prostřední příčka (G), nakonec tečka (H)

- **Reprezentace pozice**

- `byte positionMask = 0b0000LKJI;`
 - Pozice očíslováme zprava doleva logickými čísly 0 (L) až 3 (I)
 - Bit 0 (neaktivní), bit 1 (aktivní)
 - V masce lze ve skutečnosti aktivovat i více pozic najednou

- **Smazání displeje (v rámci inicializace)**

- Glyf s maskou `0b11111111` na pozicích `0b00001111`

P42: Ovládání displeje

Zobrazte vybraný glyf na vybrané pozici segmentového displeje

- Glyf bude zadán maskou
- Pozice logickým číslem

P43: Zobrazení číslic

Zobrazte vybranou číslici na dané pozici displeje

- Nejprve zkonstruujeme masky glyfů jednotlivých číslic

```
constexpr byte digitGlyphs[] = {  
    0b11000000, // 0  
    ...  
};
```

- Vložíme je do **překladačového pole** z číslic do masek



- Následně vše experimentálně otestujeme
 - Na jedné vybrané pozici zobrazíme číslici odpovídající poslední cifře hodnoty aktuálního času v sekundách

P44: Jednociferný čítač

Zobrazte hodnotu jednociferného čítače stisků tlačítka

- Čítač tedy může nabývat jen hodnot od 0 do 9
 - Aktuální hodnotu zobrazíme na jedné vybrané pozici
 - Na začátku to bude pozice 0
- Počítadlo se ovládá pomocí tlačítek takto
 - Tlačítko B1: **inkrementace** počítadla
 - Tlačítko B3: cyklická **změna pozice** (její posunutí o 1 doleva)
 - Předpokládáme jen jednoduchý stisk bez opakování

Cvičení 5: **Arduino: Displej II**

P51: Zobrazení čísel

Naprogramujte zobrazování celých čísel na segmentovém displeji

- Předpokládáme nezáporná celá čísla od 0 do 9999
 - Zobrazené číslo bude vždy zarovnané doprava
 - Zatím budeme zobrazovat i úvodní nuly
 - Např. 0025 pro číslo 25
- **Použijeme myšlenku časového multiplexu**
 - V každém běhu funkce loop aktivujeme jen jednu pozici displeje
- Rozšířený displej naprogramujeme pomocí dědičnosti
 - `class NumericDisplay : public Display { ... }`
- Volání inicializační funkce základního displeje zřetězíme
 - `Display::initialize();`

P52: Záporná čísla

Rozšiřte náš numerický displej o podporu záporných čísel

- Nově tedy budeme uvažovat čísla od -999 do 9999
 - Symbol – se zobrazí hned před první platnou číslicí
- Současně přestaneme zobrazovat **zbytečné úvodní nuly**

P53: Jednoduchý časovač

Zobrazujte na displeji aktuální čas od začátku běhu programu

- Čas uvedeme v sekundách s přesností na 1 desetinné místo
 - Např.: 0.0 nebo 12.3
 - Počet desetinných míst bude konfigurovatelný
 - Žádné nebo desetinná tečka na pozici 0 až 3
- Zobrazování tečky
 - Rozšíříme naši stávající funkci na zobrazování číslic
 - Kombinace více masek dosáhneme pomocí bitového &

P54: Rozšířené počítadlo

Zobrazte na displeji aktuální hodnotu vylepšeného počítadla

- Počítadlo může nabývat platných hodnot od -999 do 999
 - V případě přetečení se počítadlo zastaví na uvedené min / max hodnotě a dále se už zmenšovat / zvětšovat nebude
- Počítadlo budeme ovládat pomocí tlačítek
 - Tlačítka B1 a B2: **inkrementace** / **dekrementace** počítadla
 - Tlačítko B3: cyklická **změna pozice**
- Změna hodnoty probíhá vždy o +/- 1 v daném řádu
 - Tedy +/- 1, 10 nebo 100 podle aktuálně nastavené pozice
- **Aktivní pozici** označíme na displeji pomocí tečky

Cvičení 6: **Arduino: Displej III**

P61: Zobrazení znaků

Rozšiřte náš displej o podporu zobrazování vybraných znaků

- Konkrétně chceme pracovat s následujícími znaky
 - **Písmena** anglické abecedy (case-insensitive)
 - Masky glyfů jsou ve startovacím balíčku v ReCodExu
 - **Číslice** 0 až 9
 - **Mezera** _ pro jakýkoli bílý znak
 - Nějaký speciální unikátní glyf pro všechny ostatní znaky
- Předpokládejme např. následující rozhraní
 - `void showChar(char symbol, int position)`
- Užitečné funkce a triky
 - `isAlpha`, `isDigit`, `isSpace`, `isUpperCase`
 - `symbol - 'A'` apod. pro výpočet indexu do pole masek glyfů
- Nově navrženou funkcionalitu experimentálně otestujeme

P62: Zobrazení textu

Rozšířte náš displej o podporu zobrazení textového řetězce

- Předpokládáme řetězce (maximální) délky 4
 - Řetězec na displeji zarovnáme doleva
 - V případě potřeby jej tedy zprava doplníme mezerami
 - Delší řetězec ořízneme a nadbytečné znaky ignorujeme
- Opět využijeme dědičnost
 - `class TextDisplay : public Display { ... }`
- Triky pro práci s řetězci
 - `char* pointer` vs. `const char* pointer`
 - `*pointer != '\0'`
 - `*pointer++`
- Použijeme rovněž i myšlenku časového multiplexu
- Nově navrženou funkcionalitu experimentálně otestujeme

P63: Běžící text

Naprogramujte mechanismus pro zobrazování běžících zpráv

- Předpokládáme zatím jen fixně zadaný textový řetězec
 - Jeho délka může být libovolná, klidně i nulová
 - Na displeji vždy zobrazíme výřez 4 aktuálních znaků
 - Začneme jen s prvním symbolem umístěným úplně vpravo
 - Výřez pak v pravidelných intervalech posouváme doleva
 - Za koncem řetězce vždy přidáme 4 oddělovací mezery
 - Po zobrazení celé zprávy skončíme a očekáváme další vstup
- Podporovat budeme následující veřejné rozhraní
 - `void setText(const char* string);`
 - `bool finished();`
- Nově navrženou funkcionalitu experimentálně otestujeme

P64: Běžící zprávy

Rozšiřte náš mechanismus pro postupné zobrazování více zpráv

- Tyto zprávy budou definovány pomocí konstantního pole

```
▪ constexpr char* inputMessages[] = {  
    "Hello World",  
    ...  
};
```

- Zobrazovat je pak budeme postupně cyklicky