

NSWI170 – Počítačové systémy

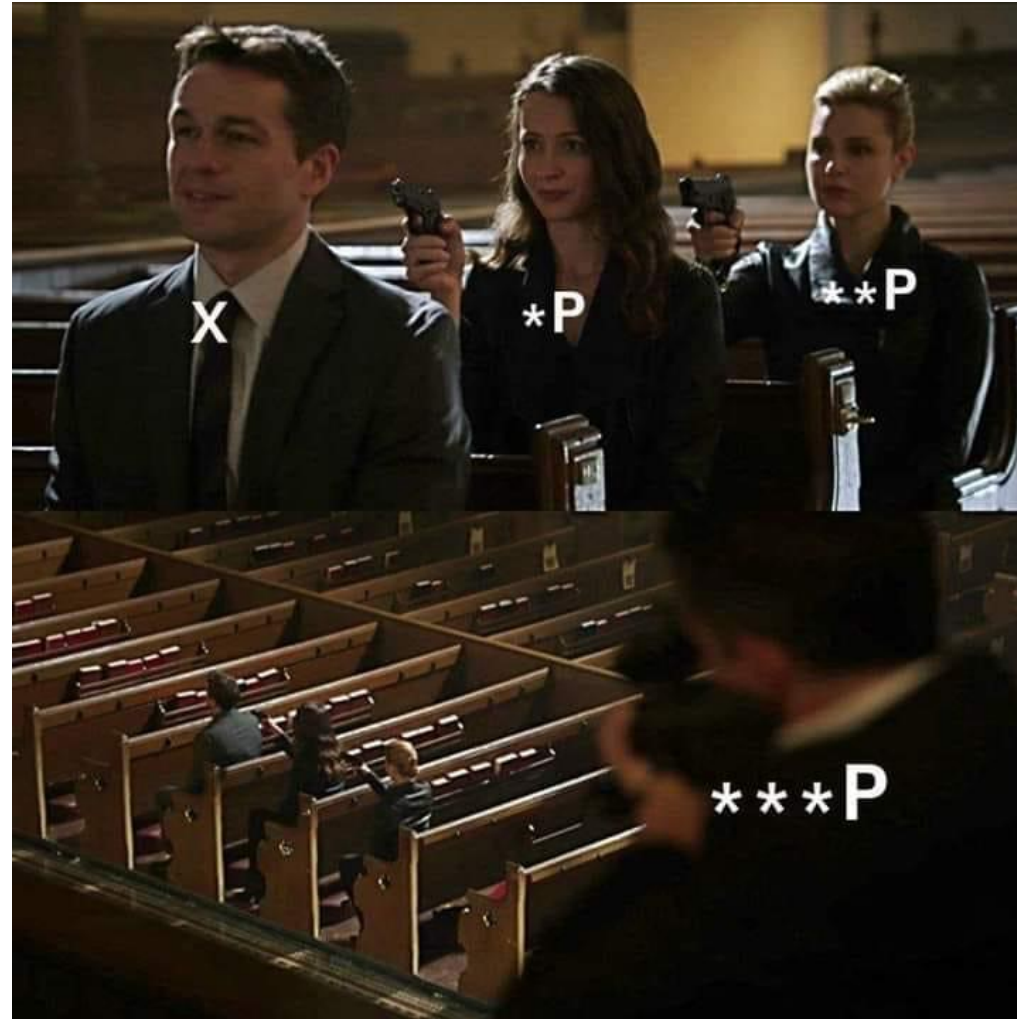
Tomáš Faltín

6. cvičení

Organizace

- Poslední „oficiální“ cvičení
 - 24.5. konzultace
- Feedback na úkoly mailem
- Zápočet
 - Odevzdané & opravené malé domácí úkoly
 - Velký domácí úkol
 - https://www.ksi.mff.cuni.cz/teaching/nswi170-web/#@tab_assignments

Ukazatele (pointers)



Ukazatele (pointers)

- Jde o typ proměnných (podobně jako int, double, pole, ...)
- `typ *jmeno`
 - Překladač kontroluje/zná typ, na který se ukazatel odkazuje
- Příklady:
 - `int *p1 = ...;` // ukazatel **p1** na typ **int**
 - `char *p2 = ...;` // ukazatel **p2** na typ **char**
 - `int **p3 = ...;` // ukazatel **p3** na typ **ukazatel na int**
- Příklady použití:
 - při dynamické alokaci (ne v Arduino)
 - malloc/new vrací ukazatele
 - práce s poli/řetězci
 - pole je vlastně ukazatel na začátek
 - výstupní parameter z funkce
 - v C se všechny parametry předávají hodnotou
 - v C++ použít reference

Ukazatele (pointers)

- Speciální operátor **&** vrací adresu proměnné

- `int x = 3;`
`int *px = &x;`

- `p_x` nyní obsahuje adresu proměnné `x`
• jinak řečeno: `p_x` „ukazuje“ na proměnnou `x`

- **Pozor** v C++ má `&` i jinou funkci – reference

- Speciální hodnota `nullptr`, když „neukazují na nic“

- `int *px = nullptr;`

- Operátor `*`

- Přístup na hodnotu uloženou na adrese, která je v pointeru

- `int x = 3;`
`int *px =`
`Serial.print(*px);`

Jak číst deklarace

- `char *(*(**foo[][8])())[]; // WTF???`
 - Použít typedef/using
 - `using my_int = int;`
 - `using int_array256_t = int[256]`

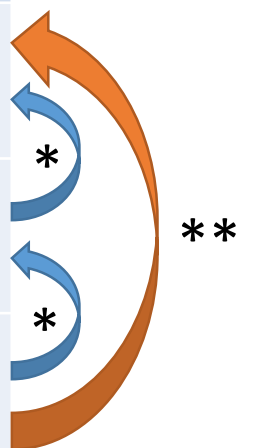
1. najít identifikátor
2. doprava (zastav se na závorce)
3. doleva (zastav se na závorce)

- Další zdroje
 - Google it („How to read declarations in C++“)
 - http://faculty.cs.niu.edu/~mcmahon/CS241/Notes/reading_declarations.html

Ukazatele (pointers) v paměti

```
int main() {  
    int i = 2;  
    int *pi = &i;  
    int **ppi = &pi;  
    Serial.print(i); // 2  
    Serial.print(pi); // 102  
    Serial.print(*pi); // 2  
    Serial.print(ppi); // 104  
    Serial.print(*ppi); // 102  
    Serial.print(**ppi); // 2  
}
```

Adresa	Obsah	(Proměnná)	Operátor *
...			
100	2	i	*
101			
102	100	pi	*
103			
104	102	ppi	*
105			
...			



Q: Co se stane pokud bychom zavolali: `Serial.print(*i)`

Q: Serial.print(*i);

```
int i = 2;  
Serial.print(*i);
```

- Program vrátí hodnotu, která leží na adrese 2, kde může ležet cokoliv ☠
- Poznámka: překladač vás to nenechá udělat, jelikož ví, že se jedná o číslo, na kterém nesmíte volat operátor *

Adresa	Obsah	(Proměnná)	Operátor *
0			
1			
2	???		
....			
100	2	i	*
101			
102	100	pi	
103			
104	102	ppi	
105			
....			

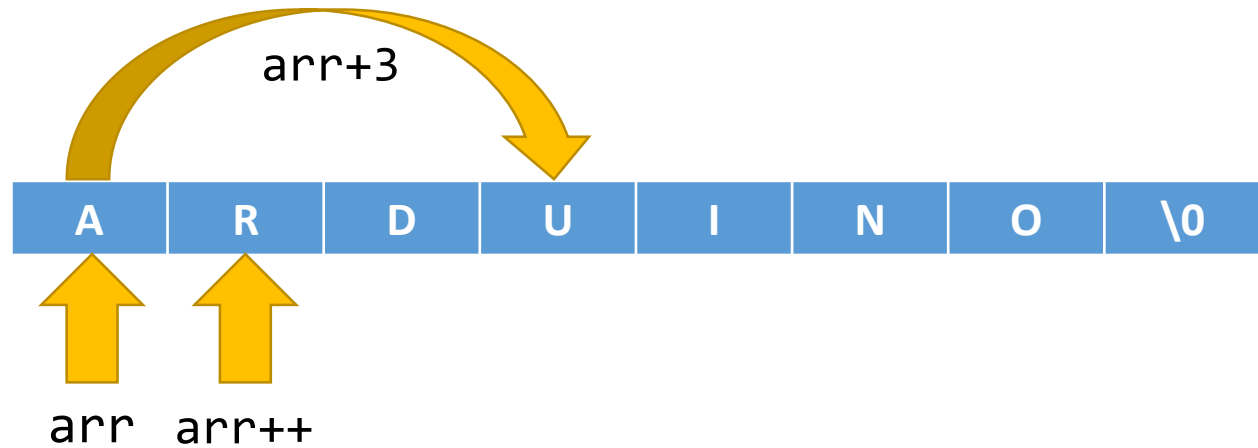
Řetězce v C

- Pole jsou (z pohledu překladače) jen zvláštní druh ukazatele, který ukazuje na první element
 - `char [] ~ char *`
 - `char arr[] = { 1, 2, 3 }; assert(arr[0] == *arr);`
- řetězec = pole znaků `char[]` zakončené znakem `'\0'`
 - Př.: `const char str*="ARDUINO"`
- Práce s řetězcí pomocí ukazatelů

A	R	D	U	I	N	O	'\0'
---	---	---	---	---	---	---	------

Práce s ukazateli

- Ukazatelová aritmetika (+, -, ++, --, ...)
 - `*arr = arr[0]`
 - `*(arr + 3) = arr[3]`
 - `++arr, arr--, *arr++`



Počítání délky řetězce

```
size_t length1(const char *str) {  
    size_t i = 0;  
    while(str[i] != '\0') { ++i; }  
    return i;  
}
```

```
size_t length2(const char *str) {  
    size_t i = 0;  
    while(*str++) { ++i; }  
    return i;  
}
```

```
int main() {  
    print(length1("Arduino"), length2("Arduino")); // 7, 7  
}
```

Any fool can write code that a computer can understand. Good programmers write code that humans can understand.

5. cvičení

Feedback

- Čím dál tím hezčí kód 😊
- Motivace
 - Velké vs. malé projekty
 - McConnell, S. "Dokonalý kód – Umění programování a techniky tvorby software"

Funny quotes

- Always code as if the guy who ends up maintaining your code will be a violent psychopath who knows where you live. (Martin Golding)
- When a programming language is created that allows programmers to program in simple English, it will be discovered that programmers cannot speak English. (Anonymous Linguist)
- Any fool can write code that a computer can understand. Good programmers write code that humans can understand.
- Any code of your own that you haven't looked at for six or more months might as well have been written by someone else. (Eagleson's law)

Pokus: delay() + segmentový displej

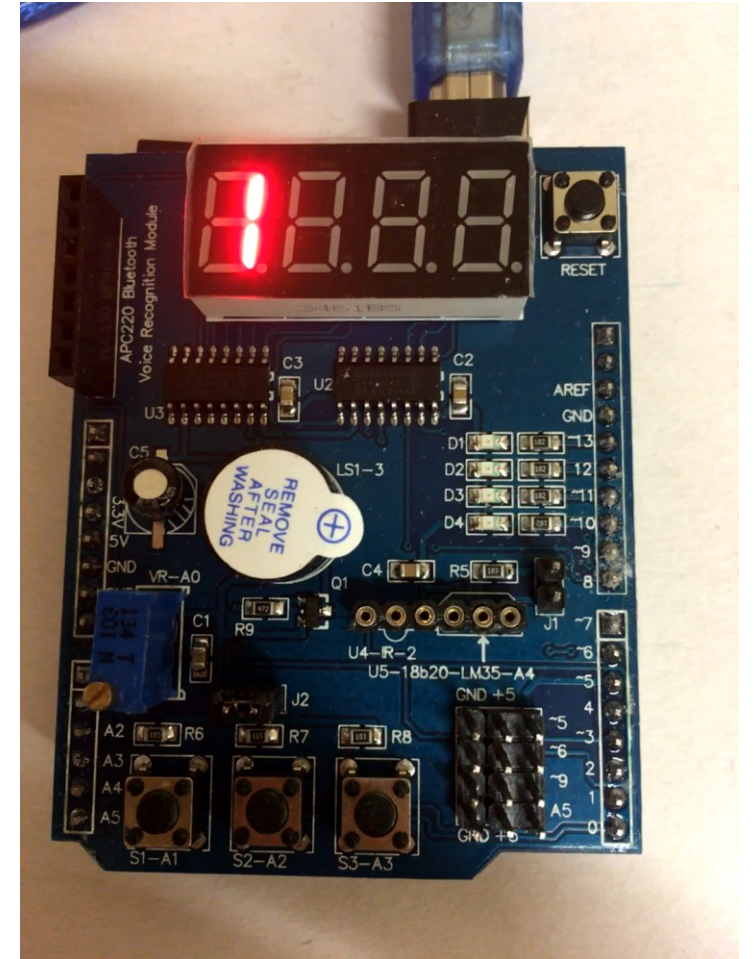
- Přidejte delay do hlavní smyčky hned za volání vaší funkce, např:

```
void loop() {  
    seg_write_number(1234);  
    delay(200);  
}
```

- Zobrazuje displej číslo správně?
- Základní (naivní) implementace na videu na dalším slide

Řešení: delay() + segmentový displej

- Časový multiplex
 - Pokud bude Arduino dostatečně rychle blikat, oko si bude myslet, že svítí
 - Můžu rychle za sebou zobrazovat znaky
 - 1 cyklus = zobrazení 1 znaku
- Zobrazení na displeji rozdělím na 2 funkce:
 1. Nastaví hodnotu
 2. V cyklu zobrazuje postupně znaky čísla



Moderní použití

- Hlavní funkce `loop()` by měla být co nejkratší, jelikož obsluhuje všechny funkce
- Složitější úlohy se podrozdělí na menší podúlohy
 - Když jsou dostatečně malé, vykonávají se v rámci jednoho volání funkce `loop()`
- Podobný princip/pattern (hlavní funkce která deleguje práci do menších funkcí) i u jiných (moderních) frameworků
 - Důraz na používání asynchronního volání
 - Např.: Node.js – JS runtime nad V8 engine (engine, který používá Chrome)



4. cvičení

Výčtový typ – enum class

- K zaznamenání stavů (více než 2 stavy)
- Nepoužívat enum

```
enum class button_state {  
    UP,  
    DOWN,  
    DEBOUNCING,  
    LONG_DOWN  
};
```

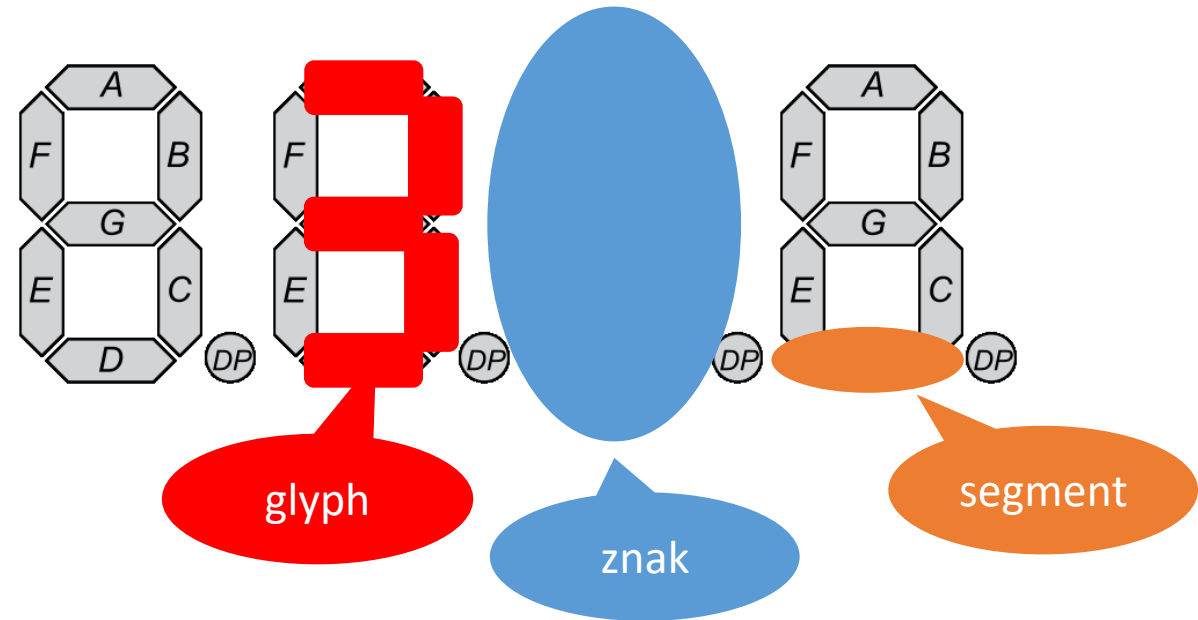
```
button_state button1_state = button_state::UP;  
  
void loop() {  
    if (button1_state == button_state::UP) {  
        // do something  
    } else if (button1_state == button_state::DOWN) {  
        // do something  
    }  
  
    switch(button1_state) {  
        case button_state::UP: {  
            // do something  
            break;  
        }  
        case button_state::DOWN:  
        case button_state::LONG_DOWN:  
            // do something  
            break;  
        default:  
            // do something  
    }  
}
```

Jak ladit Arduino (=debugging)

- Arduino IDE
 - Tools → Serial Monitor → (nastavte rychlost) 9600 baud
- `setup()`
 - `Serial.begin(9600);` // nastavte stejnou rychlost jako v IDE
- Používání
 - <https://www.arduino.cc/reference/en/language/functions/communication/serial/>
 - `print()`, `println()`
 - `Serial.print(„Hello world“); Serial.println(1234); ...`
- Vkládání ladících výpisů, kudy program běžel

Segmentový displej

- individuální ovládání segmentů
- glyph - svítící obrázek v jednom znaku



Vytvoření glyphu

- Kódování v 1 byte (8 bitů)
- Každý 1 bit odpovídá 1 segmentu

- Kódování segmentů

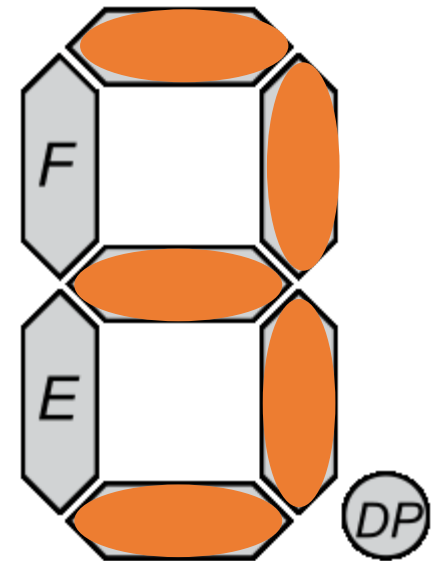
A	B	C	D	E	F	G	DP
---	---	---	---	---	---	---	----

- Inverzní logika

- LED ON = 0
 - LED OFF = 1

- Příklad: Glyph „3“

segment	A	B	C	D	E	F	G	DP
byte	0	0	0	0	1	1	0	1

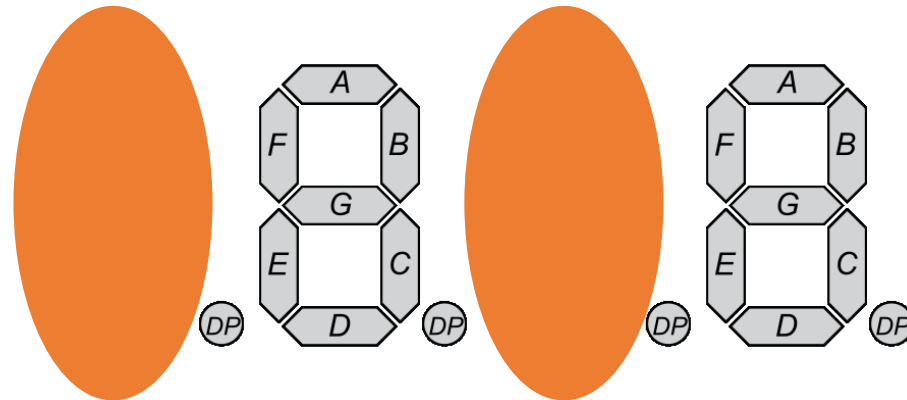
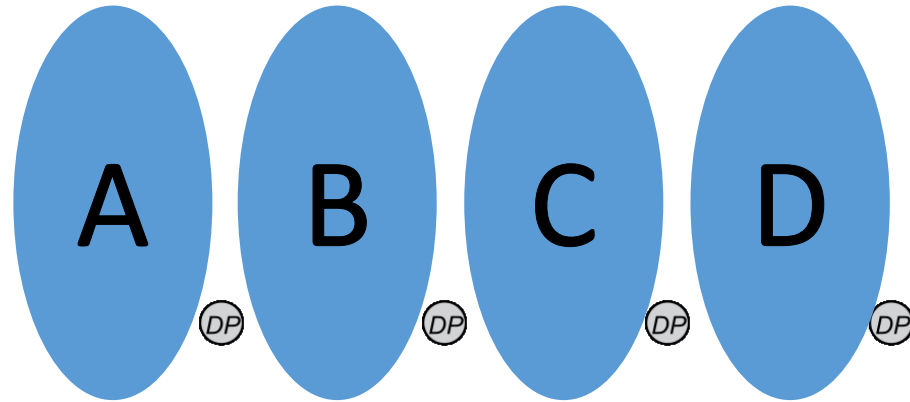


Výběr znaku

- Opět pomocí bitů
- 1 bit odpovídá jednomu znaku
 - Použity jen bity 0-3 (ostatní nezajímavé)
- Kódování

A	B	C	D				
---	---	---	---	--	--	--	--
- Příklad: aktivuj znaky A, C

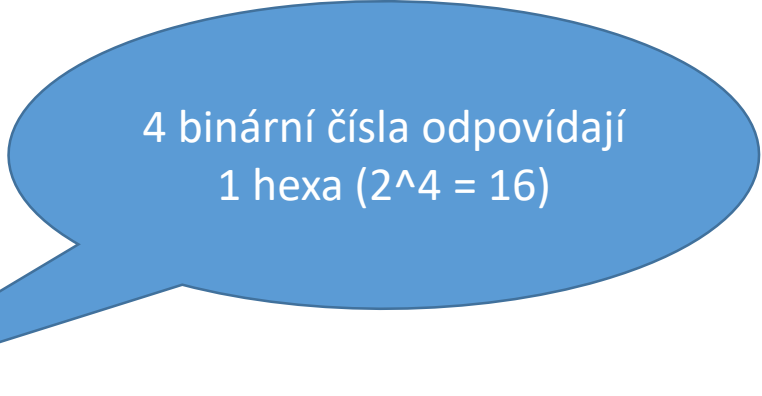
znak	A	B	C	D				
byte	1	0	0	1				



Kódování bytu

- 1 byte = 8bitů
- Zápisy
 - Binárně: 0b01101010
 - Hexadecimálně: 0x6A

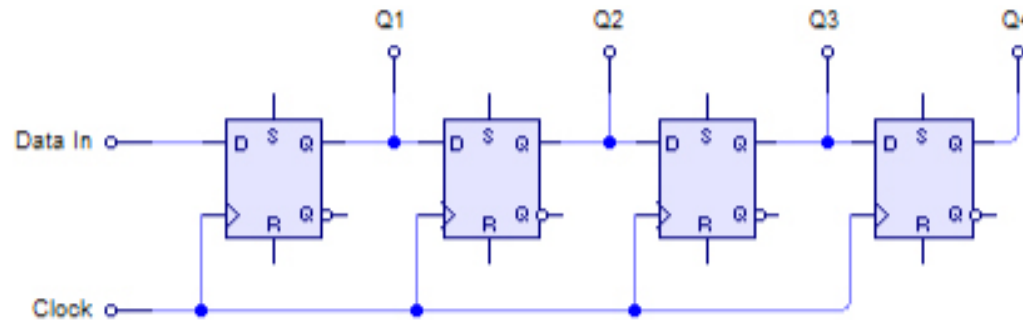
index	7	6	5	4	3	2	1	0
bin	0	1	1	0	1	0	1	0
hex	6				A			



4 binární čísla odpovídají
1 hexa ($2^4 = 16$)

Programování displeje

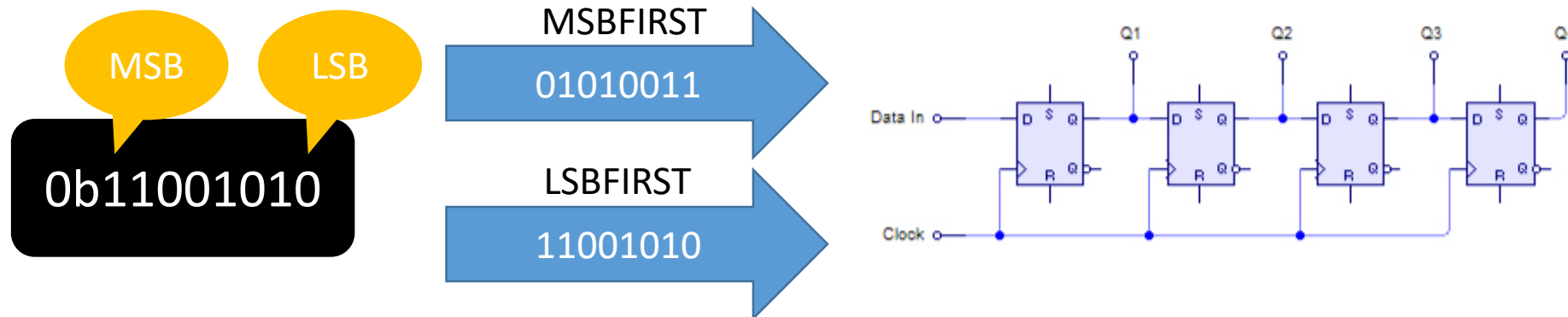
- Pomocí posuvného registru
- K ovládání slouží 3 piny
 - latch_pin: signalizace začátku/konce
 - clock_pin, data_pin: použity k posílání
- Inicializace
 - pinMode(*_pin, OUTPUT)



Programování displeje

```
void write_glyph(byte glyph, byte position) {  
    digitalWrite(latch_pin, LOW); // zavřít - začátek zápisu  
    shiftOut(data_pin, clock_pin, MSBFIRST, glyph); // pošli glyph  
    shiftOut(data_pin, clock_pin, MSBFIRST, position); // pošli pozice znaků  
    digitalWrite(latch_pin, HIGH); // otevřít - konec zápisu  
}
```

- `shiftOut(dataPin, clockPin, bitOrder, value)`
 - <https://www.arduino.cc/reference/en/language/functions/advanced-io/shiftout/>
 - bitOrder: MSBFIRST/LSBFIRST (most/least significant bit first)
 - Příklad:



Užitečné funkce pro práci s bity

- bit, bitClear, bitSet,
 - <https://www.arduino.cc/reference/en/language/functions/bits-and-bytes/bitclear/>
- & - and, | - or, ^ - xor, ...
 - $0xF5 \& 0xF3 == 0xF1$
- >> (*shift right*), << (*shift left*), ...
 - $1 \ll 2 == 4$

Programování

1. Zkus `Serial.println()`
2. Funkce na zápis glyphu – `write_glyph()`
3. Napsat číslo pomocí `MSBFIRST` a `LSBFIRST`
4. Funkci, která zobrazí libovolné číslo na displeji (DÚ)

3. cvičení

Příklad DU1 & DU2

Základní typy v C++

- Příznaky – `bool`
- Čísla
 - Znaménková – `int`, `float`, `double`
 - Nezáporná – `unsigned (int)`, `size_t`
- Znaky – `char`
- Výčtové typy – `enum class` (jindy)

Pole v C++

```
constexpr int leds[] = { led1, led2, led3, led4 };
constexpr size_t leds_size = sizeof(leds) / sizeof(leds[0]);

void print(const int array[], size_t array_length) {
    for(size_t i = 0; i < array_length; ++i) {
        printf(" %d", array[i]);
    }
}
```

Struktury v C++

```
struct S {  
    int value;  
  
    S() { // ctor - simple init of all attributes  
        value = 0;  
    }  
  
    S(int value_init) {  
        value = value_init;  
    }  
  
    int get_value() { return value; }  
};  
  
S s1; S s2(10);  
s1.attribute = 10;  
s.get_value();
```

Struktury v C++

```
struct Led {
    int pin;
    size_t turn_on_time;
    bool is_on;

    Led(int led_pin) { // ctor - simple init of data props
        pin = led_pin;
        turn_on_time = 0;
        is_on = false;
    }

    void setup() {
        pinMode(leds[i], OUTPUT);
        turn_off();
    }

    void turn_on() {
        if (!is_on) {
            digitalWrite(pin, ON);
            is_on = true;
            turn_on_time = now();
        }
    }

    void turn_off() {
        if (is_on) {
            digitalWrite(pin, OFF);
            is_on = false;
        }
    }
};
```

```
struct Leds {
    Led leds[] = {
        Led(led1_pin),
        Led(led2_pin),
        Led(led3_pin),
        Led(led4_pin),
    };

    size_t size() {
        return sizeof(leds) / sizeof(leds[0]);
    }

    void setup() {
        for(size_t i = 0; i < size(); ++i) {
            leds[i].setup();
        }
    }

    void turn_on_next() { ... }
    void turn_on_all() { ... }
    void show_binary_number(size_t number) { ... }
};

Leds leds;
constexpr size_t TIMEOUT_MS = 1000;
size_t last_turn_on;

void setup() {
    leds.setup();
    last_turn_on = now();
}

void loop() {
    if (last_turn_on + TIMEOUT_MS >= now()) {
        leds.turn_on_next();
        last_turn_on = now();
    }
}
```

Funkce - výstupní parametry

- žádný – void
- jeden – return XYZ
- více - & (reference)
 - Nedělá funkce víc věcí najednou?
 - ~~get_max_and_min()~~

```
int get_max(int array[], size_t array_length) {
    assert(array_length > 0);
    int max = array[0];
    for(size_t i = 1; i < array_length; ++i) {
        if (array[i] > max) {
            max = array[i];
        }
    }
    return max;
}

size_t get_max_index(int array[], size_t array_length,
    int &found_max_value) {
    if (array_length == 0) {
        return 0;
    }

    size_t max_index = 0;
    for(size_t i = 1; i < array_length; ++i) {
        if (array[i] > array[max_index]) {
            max_index = i;
        }
    }
    found_max_value = array[max_index];
    return max_index;
}
```

Příklad na opravu

```
int f(int a[], int b) {  
    int x = 0;  
    for(int i = 0; i < b; ++i)  
        x += a[i];  
    return sum;  
}
```

Příklad na opravu: odsazení

```
int f(int a[], int b) {  
    int x = 0;  
    for(int i = 0; i < b; ++i) {  
        x += a[i];  
    }  
    return x;  
}
```

Příklad na opravu: pojmenování proměnných

```
int f(int array[], int array_length) {  
    int sum = 0;  
    for(int i = 0; i < array_length; ++i) {  
        sum += array[i];  
    }  
    return sum;  
}
```


Příklad na opravu: typy proměnných

```
int f(int array[], size_t array_length) {  
    int sum = 0;  
    for(size_t i = 0; i < array_length; ++i) {  
        sum += array[i];  
    }  
    return sum;  
}
```

Příklad na opravu: jméno funkce

```
int sum_array(int array[], size_t array_length) {  
    int sum = 0;  
    for(size_t i = 0; i < array_length; ++i) {  
        sum += array[i];  
    }  
    return sum;  
}
```

Tlačítka

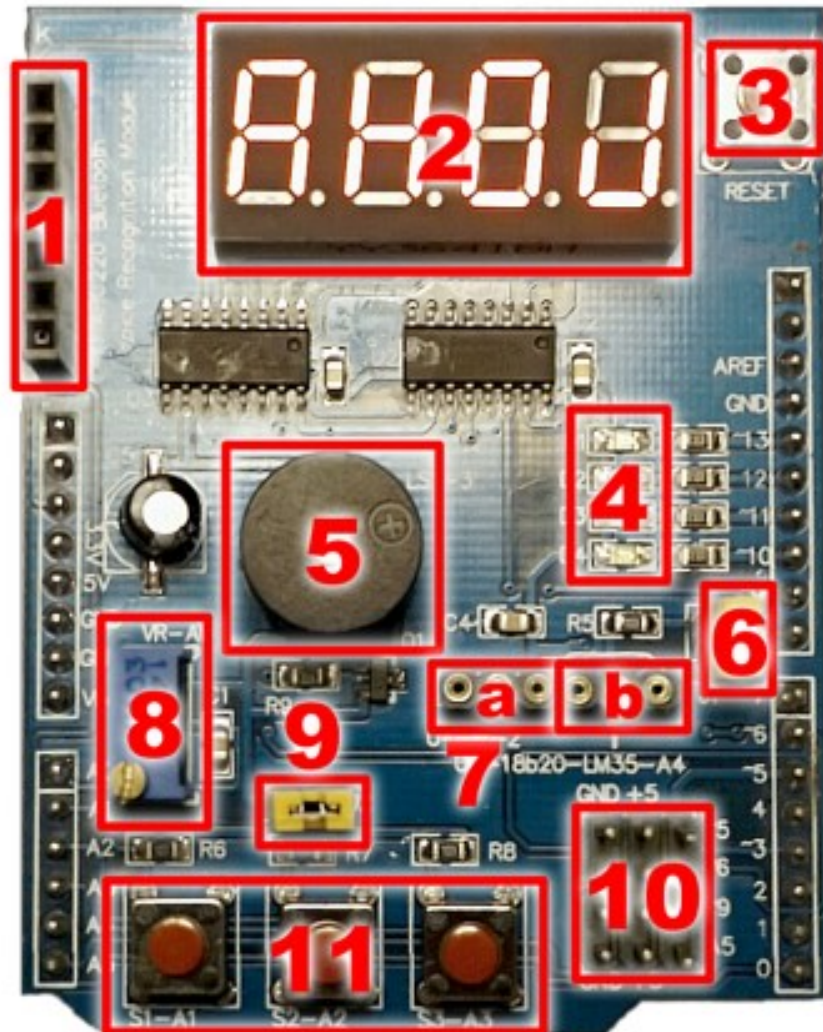
- `pinMode(button1_pin, INPUT);`
 - inicializace
- `int value = digitalRead(button1_pin);`
 - `if (!value) { ... } // tlačítko je zmačknuto`
- Odkazy:
 - <https://www.arduino.cc/reference/en/language/functions/digital-io/digitalread/>

2. cvičení

Arduino

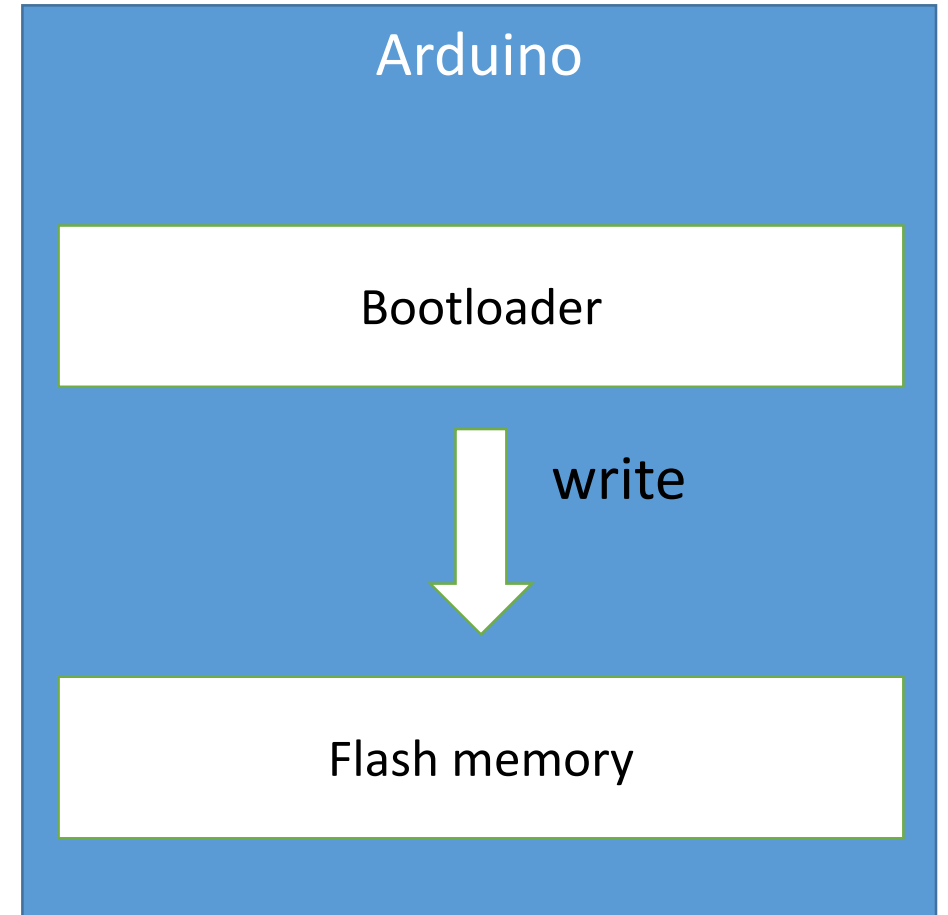
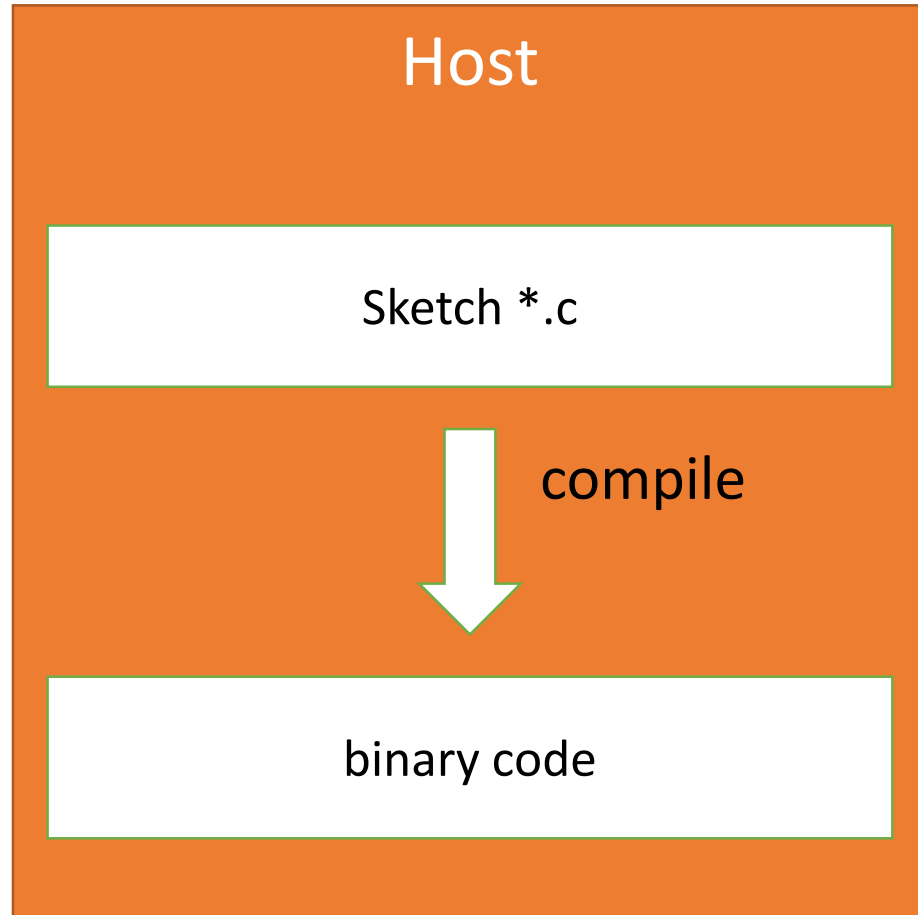
- Open-source HW SW projekt
- Arduino board + expansion board (shield)
- Arduino IDE

Arduino HW



- (1) Konektor pro Bluetooth
- (2) LED
- (3) Reset
- (4) Signalizační LED
- (5) Piezo-bzučák
- (6), (9) Propojka
- (7) Konektor pro IR
- (8) Potenciometr
- (10) Konektor pro čidla
- (11) Vstupní tlačítka

Kompilace



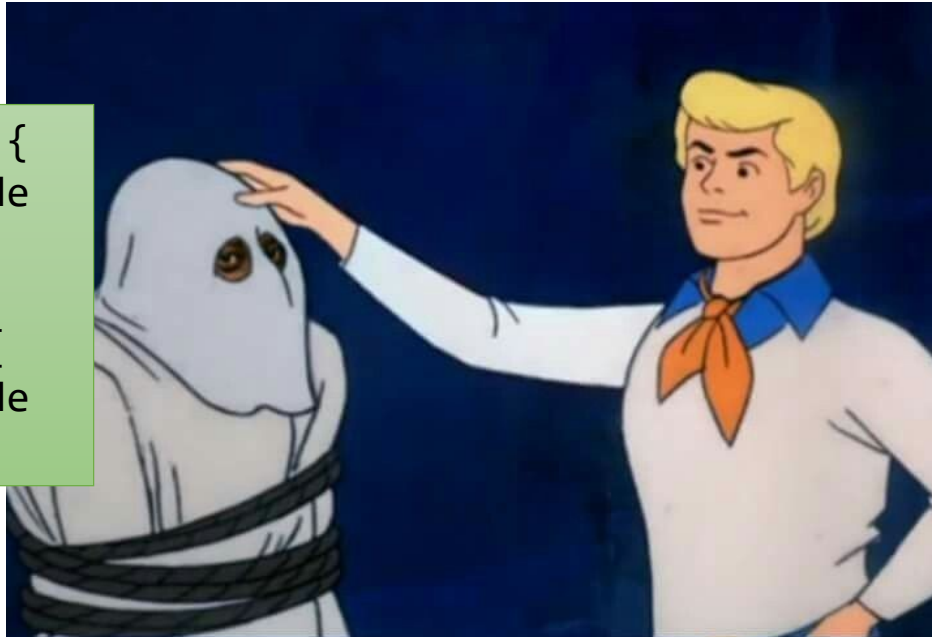
Arduino IDE

```
void setup() {  
    // put your setup code here, to run once:  
}
```

```
void loop() {  
    // put your main code here, to run repeatedly:  
    // called ~1000/s  
}
```



```
void setup() {  
  // init code  
}  
  
void loop() {  
  // main code  
}
```



```
int main() {  
  setup();  
  while(true) {  
    loop();  
  }  
}
```



Dobré programátorské zvyky

- Zapnout warnings překladače
- Nepoužívat copy&paste
 - Funkce, pole, ...
- Používat konstanty
 - `#include "funshield.h"`

Úkoly

1. Inicializovat LEDky
 - `pinMode(pin, OUTPUT/INPUT)`
2. Blikat vybranou LEDkou
 - `digitalWrite(pin, HIGH/LOW)`
 - `delay(ms)`
3. Blikat všemi ledkami najednou
 - ne C&P (co kdyby LEDek bylo 1M)
4. Blikat bez delay
 - `millis(ms)`
5. Had délky 2
6. Had libovolné délky

Odkazy

- https://www.ksi.mff.cuni.cz/teaching/nswi170-web/#@tab_links

1. cvičení

Komunikace

- Bud'te proaktivní
- Web
 - https://fan1x.github.io/computer_systems.html
 - <https://www.ksi.mff.cuni.cz/teaching/nswi170-web/>
- Mattermost
- Mail
- Zoom 😊

Průběh cvičení

- Účel předmětu NSW170 – Počítačové systémy
 - Vysvětlit, co informatik potřebuje vědět o hardware a systémovém software
 - Seznámit se s jazykem, který je pravým opakem Pythonu
 - Vyzkoušet si programování v těsném kontaktu s hardware
- Obsah přednášky (Jakub Yaghob nebo Lubomír Bulej)
 - 1..2 – základy jazyka C
 - 3..14 – operační systémy, překladače, ...
- Obsah cvičení
 - Předmět je sice 2/2, ale cvičení je pouze jednou za 14 dní
 - Druhou dvouhodinu strávíte u domácích úkolů (a vaši učitelé při jejich kontrole)
 - 1 – první kroky v C++
 - 2..6 – programování pro Arduino
- Od třetího týdne přednáška se cvičením nesouvisí
 - Ani zápočet se zkouškou

Zápočet

- Před druhým cvičením (22.3.) si zajistěte prostředí k práci
 - **Arduino (v knihovně nebo koupit on-line)**
 - Nainstalujte si na vašem počítači **Arduino IDE** (pro řešení domácích úkolů)
- Na cvičení budou zadávány úlohy
 - Odevzdání do ReCodexu
 - **1 týden na řešení**
 - Arduinovské úlohy na sebe navazují, řešení tedy budete sami potřebovat
- Na šestém cvičení bude zadána hlavní domácí úloha

IDE

- V čem tedy budete programovat?
 - Technicky to bude C++
 - C++ je (téměř) nadmnožina C
 - U některých C-konstrukcí má C++ o něco přísnější pravidla, tím včas odhalíte některé chyby
 - Půjčíme si z C++ několik drobností usnadňujících život
 - Parametry předávané odkazem, prázdné závorky v deklaraci funkce bez parametrů, ...
 - Složitější vlastnosti C++ nejsou v nízkoúrovňovém prostředí příliš užitečné
 - Často ani nejsou dostupné kvůli omezené kapacitě hardware
- Kde?
 - 1. cvičení: coliru.stacked-crooked.com
 - Webový editor schopný zkompileovat a spustit jednoduchý program v C++
 - Kdo to umí, může používat jakýkoliv jiný editor a překladač C++
 - Zbytek cvičení: Arduino IDE - www.arduino.cc/en/main/software
 - Aplikace pro Windows/Linux/macOS
 - Editor, překladač, dálkový (USB) ovladač Arduina

Čas na hraní 😊

Hello World

```
#include <stdio>
```

```
int main()  
{  
    printf("Hello World :)\n");  
}
```

Tajemná funkce 1

```
#include <stdio>
```

```
int fn1(int array[], int length) {  
    int res = 0;  
    for(int i = 0; i < length; ++i) {  
        res += array[i];  
    }  
    return res;  
}
```

```
int main()  
{  
    int array[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
    int res = fn1(array, 9);  
    printf("Result: %d", res);  
}
```

Tajemná funkce 2

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

```
int fn2(int array[], int length, int number) {  
    int i = 0;  
    while(i < length && array[i] != number) {  
        ++i;  
    }  
  
    return i;  
}
```

```
int main()  
{  
    int array[] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10};  
    static constexpr int SIZE = 100;  
    int res = fn2(array, SIZE, 4);  
    printf("Result: %d", res);  
}
```

Tajemná funkce 3 (1/2)

```
#include <stdio.h>

int fn3(int array[], int length) {
    int j = 0;
    int k = 0;
    for(int i = 0; i < length; ++i) {
        if (array[i] % 2 == 0) {
            ++j;
        } else {
            ++k;
        }
    }

    if (j > k) {
        return j;
    } else if (k > j) {
        return -k;
    } else {
        return 0;
    }
}

int main()
{
    int array[] = {1, 2, 3, 4, 5, 6, 7, 8, 9};
    static constexpr int SIZE = 9;
    int res = fn3(array, SIZE);
    printf("Result: %d", res);
}
```

Tajemná funkce 3 (2/2)

```
int count_and_compare_odd_even(int array[], int length) {  
    int even_count = 0;  
    int odd_count = 0;  
  
    for(int i = 0; i < length; ++i) {  
        if (array[i] % 2 == 0) {  
            ++even_count;  
        } else {  
            ++odd_count;  
        }  
    }  
  
    if (even_count > odd_count) {  
        return even_count;  
    } else if (odd_count > even_count) {  
        return -odd_count;  
    } else {  
        return 0;  
    }  
}
```

Úkoly

1. Hello World
2. Nakreslit trojúhelník
3. Nakreslit vánoční stromeček
4. Vypsát průměr hodnot v poli
5. Nakreslit graf hodnot v poli
6. Nakreslit klouzavý průměr hodnot v poli
 - a) Pro fixní N
 - b) Obecně pro N po sobě jdoucích hodnot
7. Histogram

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

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

