

# Počítačové systémy

---

Adam Šmelko

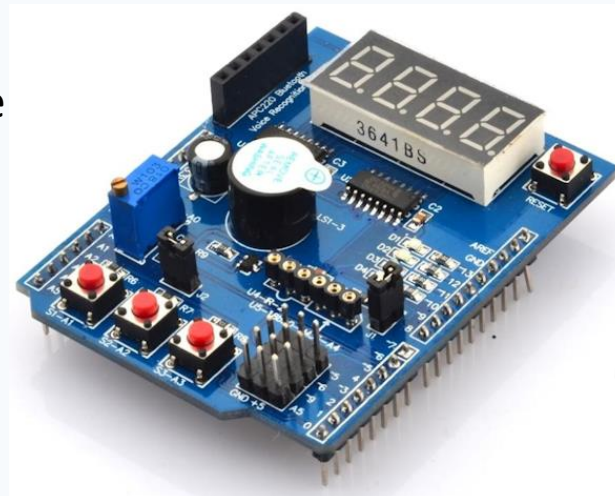
[smelko@d3s.mff.cuni.cz](mailto:smelko@d3s.mff.cuni.cz)

NSWI170 - cvičení

LS 2021/2022

# Organizace cvičení

- Účel předmětu NSW1170 – 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 Martin Kruliš)
  - 1..2 – základy jazyka C/C++
  - 3..14 – operační systémy, překladače, ...
- Obsah cvičení
  - Cvičení je pouze jednou za 14 dní
    - Druhou dvouhodinu strávíte u domácích úkolů (a vaši učitelé při jejich kontrole)
  - 1 – první kroky v C++ - repl.it
  - 2..7 – programování pro Arduino
- Od třetího týdne přednáška se cvičením nesouvisí
  - Ani zápočet se zkouškou



# Zápočet

- Podmínky získání zápočtu
  - Před druhým cvičením si zajistěte prostředí k práci
    - Sežene si Arduino + příslušný shield
      - Lze vypůjčit v malostranské knihovně MFF po předchozí domluvě
      - Lze koupit v e-shopu
    - Nainstalujte si na vašem počítači Arduino IDE (pro řešení domácích úkolů)
      - Budete-li chtít na cvičeních pracovat na svém notebooku, ověřte **před** druhým cvičením podle návodu, že vaše Arduino IDE funguje včetně spojení s Arduinem
  - Na prvním až šestém cvičení budou zadávány úlohy
    - Zdatní jedinci je možná zvládnou na místě, ostatní dodělají doma
    - Finální řešení těchto úloh submitujte do Recodexu
    - **Nejpozději týden po zadání, druhý týden je bude cvičící připomínkovat**
    - Arduinovské úlohy na sebe navazují, řešení tedy budete sami potřebovat
  - Na posledním cvičení bude zadána hlavní domácí úloha
    - Její řešení rozhodne o úspěchu v předmětu

- 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: [repl.it](https://repl.it)
    - 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](https://www.arduino.cc/en/main/software)
    - Aplikace pro Windows/Linux/macOS
    - Editor, překladač, dálkový (USB) ovladač Arduina

# Jazyk C

---

a troška C++

- Čím se liší C(++) od Pythonu?
  - Na první pohled
    - Syntaxe – složené závorky místo indentace, ++, for(;;), ...
    - Staticky typovaný jazyk – vše musí mít deklarovaný typ
    - Jiné názvy a rozhraní kontejnerů a dalších knihovných záležitostí
  - Na druhý pohled
    - tímhle se dnes C(++) liší od všeho živého kromě Fortranu
    - Explicitní dynamická alokace/dealokace - C(++) nemá garbage-collector
      - Dynamicky alokovat ale stejně nebudeme
    - Proměnné obsahují objekty, ne odkazy na ně
      - U imutabilních typů (čísla) se to pozná jen na rychlosti
      - U polí a řetězců to začíná být zajímavé
      - Chcete-li odkaz (ukazatel, referenci, ...), musíte to explicitně deklarovat
  - Noční můry (nejen začínajících) programátorů v C
    - Téměř žádné běhové kontroly, lokální proměnné nejsou automaticky inicializovány
      - Spektakulární způsoby pádu vadných programů (+ tisíce exploitů pro hackery)
    - Ukazatele na lokální proměnné, přetypování a další low-level triky
  - Python je limuzína s řidičem, C je jízda na motorce (bez helmy)

# Python C

```
def hello(p):  
    print("Hello, number {}".format(p))
```

```
# this is the main block
```

```
hello(7)
```

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

- stdio.h obsahuje printf

```
void hello(int p)
```

```
{
```

```
    printf("Hello, number %d\n", p);
```

- %d očekává další argument typu int
- Když tam nebude, budou se dít podivné věci

```
}
```

- Funkce main je vyvolána po startu programu a inicializaci C-knihoven

```
int main()
```

```
{
```

```
    hello(7);
```

```
    return 0;
```

```
}
```

- Hodnota vrácená z main se předá tomu, kdo tento program vyvolal (ten ji obvykle ignoruje)

# Python C

```
def hello2(p):
```

```
    print('Hello, number', end='')
```

- Pojmenované (tj. ne poziční) argumenty ve volání funkce nemají v C ekvivalent
- Konstanta v apostrofech je v Pythonu řetězec stejně jako v uvozovkách. V jazyce C to neplatí.

```
    print(p)
```

```
# this is the main block
```

```
hello2(7)
```

```
hello2("7")
```

- Přestože 7 a "7" jsou v Pythonu objekty různého typu, mohou být oba drženy proměnnou p
- Odlišnost typů se zde projevuje pouze jinou implementací uvnitř funkce print (která je pro řetězec jednodušší než pro číslo)

- Statická typová kontrola znemožňuje obraty, které jsou v dynamických jazycích běžné

```
void hello2(int p)
```

```
{
```

```
    printf("Hello, number");
```

```
    printf(p);
```

- Tento řádek překladač odmítne – printf očekává řetězec a automatická konverze čísla na řetězec v C ani C++ neexistuje

```
    printf("Hello, number " + p + "\n");
```

- Ani tohle neprojde – operátor '+' pro zřetězení sice v C++ existuje, ale neakceptuje parametry číselných typů. '+' navíc pracuje s C++ verzí řetězců (typ std::string) – tyto řetězcové konstanty (i formální argument funkce printf) jsou ale C verze (typ char[])

```
}
```

```
int main()
```

```
{
```

```
    hello2(7);
```

```
    hello2("7");
```

- Tento řádek překladač odmítne – funkce hello2 očekává int, automatická konverze řetězce na číslo v C ani C++ není

```
    return 0;
```

```
}
```



- Co dělá tento program v Pythonu?

```
a = [1,2,3,4,5,6,7,8,9,10]
```

```
s = 0
```

```
i = 0
```

```
while i < 10:
```

```
    s += a[i]
```

```
    i += 1
```

```
print(s)
```

- Co dělá tento program v Pythonu?

```
a = [1,2,3,4,5,6,7,8,9,10]
s = 0
i = 0
while i < 10:
    s += a[i]
    i += 1
print(s)
```

- Vypisuje 55

- Součet čísel v poli
  - Opravdový Pythonista by to samozřejmě napsal jinak

- Co dělá tento program v C?

```
int main()
{
    int a[10] = {1,2,3,4,5,6,7,8,9,10};
    int s = 0;
    int i = 0;
    while (i < 10)
        s += a[i];
        i += 1;
    printf("%d\n", s);
}
```

- Co dělá tento program v C?

```
int main()
{
    int a[10] = {1,2,3,4,5,6,7,8,9,10};
    int s = 0;
    int i = 0;
    while (i < 10)
        s += a[i];
        i += 1;
    printf("%d\n", s);
}
```

- Cyklí navždy

```
int main()
{
    int a[10] = {1,2,3,4,5,6,7,8,9,10};
    int s = 0;
    int i = 0;
    while (i < 10)
    {
        s += a[i];
        i += 1;
    }
    printf("%d\n", s);
}
```

- Pythonisti v C: Vždy pište složené závorky za if/while/for
  - ikdyby tam zatím byl jen jeden příkaz

```
int main()
{
    int a[10] = {1,2,3,4,5,6,7,8,9,10};
    int s = 0;
    int i = 0;
    while (i < 10)
    {
        s += a[i];
        ++i;
    }
    printf("%d\n", s);
}
```

- For-cyklus je jen zkratka za while s inicializací a aktualizací

```
int main()
{
    int a[10] = {1,2,3,4,5,6,7,8,9,10};
    int s = 0;
    for (int i = 0; i < 10; ++i)
    {
        s += a[i];
    }
    printf("%d\n", s);
}
```

# Konštanty

- Číslo vyskytující se v programu víckrát je poukázka na problém

```
int main()
{
    // deset je moc, devet staci
    int a[9] = {1,2,3,4,5,6,7,8,9};
    int s = 0;
    for (int i = 0; i < 10; ++i)
    {
        s += a[i];
    }
    printf("%d\n", s);
}
```

- Pojmenujte všechny konštanty
  - Kromě 0 a 1

```
int main()
{
    // deset je moc, devet staci
    const int N = 9;
    int a[N] = {1,2,3,4,5,6,7,8,9};
    int s = 0;
    for (int i = 0; i < N; ++i)
    {
        s += a[i];
    }
    printf("%d\n", s);
}
```

- Tuto formu hlavičky for-cyklu si zapamatujte – je nejčastější

# Výčtové typy

- Číslo vyskytující se v programu  
víckrát je poukázka na problém

```
int get_something() { /*...*/ }
void wait_for_something() {
    int state = 0;
    for (;;) {
        int x = get_something();
        if ( state == 0 && x == 2 ) {
            state = 2;
        } else if ( state == 1 && x == 0 ) {
            break;
        } else if ( x == 1 ) {
            state = 1;
        }
    }
}
```

- Jak dlouho si budete pamatovat, co  
ta čísla znamenají
- Totéž číslo navíc označuje různé věci

- Používejte výčtové typy

```
enum class input_t { LOW, MED, HIGH };
enum class state_t { INIT, FINAL, OPEN };
```

```
input_t get_something() { /*...*/ }
void wait_for_something() {
    state_t state = state_t::INIT;
    for (;;) {
        input_t x = get_something();
        if (state == state_t::INIT && x == input_t::HIGH) {
            state = state_t::OPEN;
        } else if (state == state_t::FINAL && x == input_t::LOW) {
            break;
        } else if (x == input_t::MED) {
            state = state_t::FINAL;
        }
    }
}
```

# Hello world

- jazyk C
  - kompilovaný, staticky typovaný
  - blízko hw

- ➡ hello world ([link](#) ➡ fork)
- ➡ 1.1 funkce stars, která vypíše n hvězdiček na řádek

void stars(int n) {....}

- ➡ 1.2 trojúhelník z hvězdiček velikosti n

Použijte předchozí funkci

DEKOMPOZICE

- ➡ 1.3 vánoční stromeček z hvězdiček

- rozšíření:
  - 1.3b na každé m-té úrovni ozdoba
  - 1.3c košatý strom

funkce

start programu

cyklus

životnost i v bloku

```
#include <stdio.h>

void hello()
{
    printf("Hello World\n");
}

int main()
{
    hello();
}
```

```
for (int i=0; i<n; ++i) {
    printf("*");
}
```

```
if (n < 10) {
    ....
} else if (m < 20) {
    ....
} else {
    ....
}

while (n < 10) {
    ....
}
```

1.2

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

1.3

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

1.3b

```
      *
    ***
  *****
0 ***** 0
  *****
  *****
  *****
```

1.3c

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

# Průměr pole

- ➡ průměr hodnot pole
- hvězdičkami
- vypište tolik hvězdiček, kolik je průměrná hodnota

všechny proměnné musejí být deklarované

typ návratové hodnoty funkce

formální parametr

lokální proměnná

typ

```
int f(int y) {  
    int z;  
    return y+z;  
}  
  
int main() {  
    int x = f(1);  
}
```

návratová hodnota

skutečný parametr

pole

inicializované pole

index  
vždy 0..n-1

```
int pole[10];  
int a[] = { 0, 1, 2, 3, 4 };  
+ - * / % < <= == != >= >  
  
/* multiline comment */  
// comment
```

libovolně velké pole

```
int avg(int pole[], int count) {  
    sizeof(pole)  
    ....  
}  
  
int main() {  
    int a[] { 10, 12, 14, 16 };  
    int a_cnt = sizeof(a)/sizeof(a[0]);  
    int x = avg( a, a_cnt);  
    stars( x);  
}
```

překladač  
nezná velikost

velikost  
celého pole

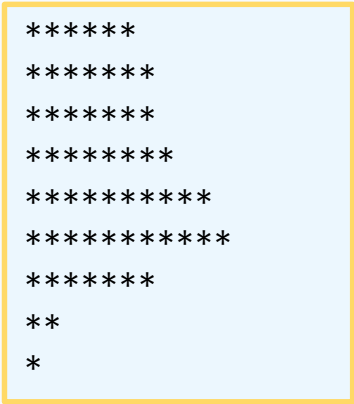
velikost  
jednoho prvku



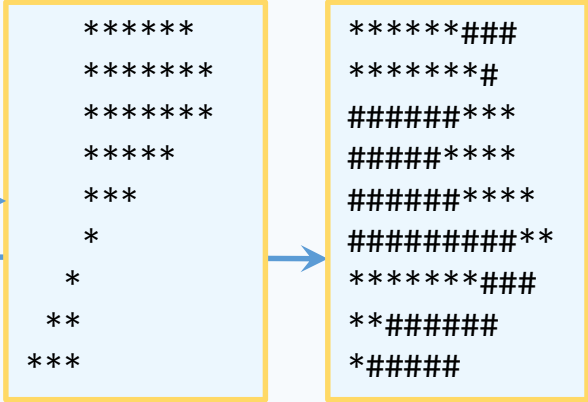
# Klouzavý průměr naměřených hodnot

- ➡ 1.4 teploměr
  - v pravidelných intervalech teplota
  - 1.4a graf (hvězdičky) hodnot
  - 1.4b graf klouzavého průměru hodnot
    - velikost klouzání - parametr

0	1	2	3	4	5	6	7
8	9	11	13	14	14	12	9



- další varianty
  - ♥ horizontální časová osa
  - 1.4c i záporná čísla (záporné ◀■, kladné ■▶)
  - ♥ dvě pole ~> prolínající se grafy (\*,#)
    - větší hodnota je "vespod"



- ➡ 1.5 nespolehlivý teploměr
  - některé záznamy neplatné
    - speciální hodnota
      - = předchozí hodnota
  - ♥ lineární aproximace

```
const int no_value = -999;  
int teploty[] { 10, 12, no_value, no_value, 20 };
```

♥ ≡ pro zájemce, na hraní ...

# Odevzdání

- V Recodexu nová uloha **Ceplomeris**

- vypracovat do týdne
- osobní připomínky
  - opravit před následujícím cvičením
  - reupload do Recodexu

- půjčit/sehnat Arduino

- zkompletovat
- opatrně - nesahat na piny!

- instalace Arduino IDE

- vyzkoušet propojení
- ❤️ hrát si

- Dotazy => Mattermost

- **nswi170-compsys-smelko**

```
#include <stdio.h>

constexpr int no_value = -999;
int temperatures[] { .... };

int main()
{
    ....
}
```