Úvod

NSWI170

Patrik Dokoupil

Úvod

- Podmínky zápočtu
  1. **Včas** splnit 6 úloh zadaných na cvičení
     - 7 dní na vypracování (**deadline** vždy další úterý ve 12:20)
     - Správnost 100% testů v ReCodExu
  2. Získat alespoň 7 bonusových bodů z úloh (viz další slide)
  3. Úspěšně splnit závěrečný test v labu (až po splnění 1. a 2.)
- Hlavním cílem cvičení je **naučit se obecné zásady programování**
  - Detailní popis a syntaxe jazyka C bude spíše na přednáškách, nikoliv na cvičení
  - Arduino
  - **5 pravidel (**„Unforgivable curses “**)** jejichž dodržování bude vyžadováno (viz Coding Guidelines)
  - **Ethical Guidelines**: odevzdaný kód musí být psán Vámi (tj. nikoliv AI nástroji / někým jiným)
- Komunikace -> mattermost (pozvánka v SISu)

Průběh cvičení a úloh

- Na každém cvičení bude zadána série úloh kterou se postupně dostaneme až ke „hlavní úloze“ kterou je nutné odevzdat.
- Pokud hlavní úlohu nestihnete na cvičení, je nutné ji dodělat doma.
- **Bonusové body**: u odevzdaných úloh udělám review
  - Buď je akceptuji (dostane bonusové body)
  - Nebo doplním komentáře co opravit a dostanete možnost opravené řešení odevzdat znovu (za určitých okolnosti lze i více iterací, ale není pravidlem)
    - Opravujte **všechny výskyty** chyb
  - Pro zisk bonusových bodů je nutné opravené řešení odevzdat **do 4 týdnů od zadání** dané úlohy
  - Za N-tou úlohu je N bonusových bodů
- Zadáno bude vždy také jedno pravidlo, dodržení pravidla *i* se vyžaduje od úlohy *i+1*.
  - Opakované nedodržení pravidel u vyšších úloh **může vést k neudělení zápočtu**
  - Nedodržení v závěrečném testu **vede k neudělení zápočtu!**
- Další informace na webu

Jazyk C/C++

NSWI170: Lab 01

Patrik Dokoupil

Credit: Martin Kruliš

Zopakování základů jazyka, rozdíly od Pythonu

- Staticky typovaný jazyk
  - Vše musí mít deklarovaný typ (proměnné, parametry, návratové hodnoty, …)
- Jiná syntaxe:

Python

```
import module

# main blok

def func1(p): # Návratový typ/typ parametru není deklarován
   print('Hello from func1: ', p) # Jednoduché uvozoky jsou string

def func2(a, b):
   s = 0
   for i in range(a, b):
      if is_prime(i): # Jiný syntax pro for + chybí závorky v podmínce
         s += i
   return s

# Stejnou funkci voláme s jiným typem parametru
func1(5)
func1('some text')

# Pojmenovaný parametr
func2(1,b=2)
```

C/C++

```
#include<stdio.h> // hlavičkové soubory, ne moduly

void func1(int p)  { // návratový typ void (nic), parametr int
   printf("Hello from func1: %d\n", p); // \n newline, tj nový řádek, ""
}

int func2(int a, int b) { // návratová hodnota i parametry mají opět typ
   int s = 0; //lokální proměnná má taky typ
   for (int i = a; i < b; ++i) { // odlišná syntaxe for loopu, ++increment
      if (is_prime(i)) {// závorky kolem podmínky
         s += 1;
      }
   }
   return s;
}

int main() {
   func1(5);
   func1("some text"); // selže, špatný typ
   func2(1, 2); // nelze pojmenovat parametry
   return 0;
}
```

; za každým příkazem

Bloky jsou určeny { }, ne odsazením! Pro čitelnost však ideálně stejně odsazujeme

Pravidlo č. 1

- #1 Dekompozice kódu
  - Rozdělujte kód na menší, logické a znovupoužitelné celky pomocí **funkcí**
    - Pokud je kus kódu použit na více místech (vyhněte se copy&paste)
    - Pokud se hodí kus kódu pojmenovat (je to nějaký logický celek/algoritmus apod.)
    - Pokud je souvislý kus kódu příliš dlouhý
    - Pokud se chcete vyhnout kolizím jmen proměnných
  - Časté chyby kterým se vyhnout
    - Kód je ve finále znatelně delší než byl
    - Funkce očekává velkou spoustu parametrů/modifikuje globální proměnné
    - Nenapadá Vás smysluplný název funkce
  - Dobrá funkce dělá jen jednu věc a ta jasně plyne z jejího názvu
    - Pure funkce – nemá žádné vedlejší efekty, pracují pouze s argumenty

Cvičení

- T1: Hello World
  - Visual Studio (nebo libovolný jiný editor + kompilátor / https://repl.it/)
  - `printf("string");`
  - Vypište “Hello world”
- T2: Napište si první funkci
  - Které na vstupu bere integer a vypíše tolik * na řádce
  - `void function_name(int n) { /* body */ }`
- T3: Vypište trojúhelník s výškou 5
  - Pomocí již definovaných funkcí
- T4: Vypište trojúhelník s výškou 42

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

Cvičení

- T5: Napište funkci která vypíše vycentrovaný trojúhelník dané výšky
  - Snažte se využít funkce které již máte

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

  - Extension 5.1: upravte funkci tak, aby nakreslila lichoběžník (tj. vynechat prvních k řádek trojúhelníku)

Cvičení*

- Extension 5.2: Nakreslit vánoční stromeček
  - Výška segmentu je parametr s
  - První segment začíná 1 *
  - Každý další začíná s o 2 více * než ten předchozí
  - Každých o řádek nakreslete ornament
    - Nějaký hezký symbol na obou stranách
    - Parametr o je zcela nezávislý na s
  - Př. $s = 3, o = 4$

```
          *
         ***
        *****
       o***o
        *****
       *******
        *****
      o*******o
     *********
       *******
     *********
   o***********o
```

Pole

- Sekvence hodnot stejného typu
- K prvkům přistupujeme pomocí indexu (zero based, tj. začíná od nuly)

```
int x[42]; // Pole s pevně zadanou délkou
int y[] = { 1, 2, 3, 4, 5 }; // Pole s pevnou, auto-určenou délkou
int z[2][4] = { { 1, 2, 3, 4 }, { 10, 20, 30, 40 } }; // Matice 2x4
y[1] = z[0][2]; // Indexování polí
```

- Pozor, že pole si nepamatuje svoji délku, pokud t

```
int avg(int a[], int count) {
    …
}
```

Pole libovolné délky

Potřebujeme tedy předat i jeho délku

Cvičení

- T6: Napište funkci která zobrazí graf
  - Hodnoty (typ integer) jsou v poli a každá hodnota se vytiskne jako odpovídající počet * na řádku (x=5 -> *****)
  - Zkušební vstup

```
int temps[] = { 3, 4, 4, 5, 6, 8, 11, 11, 10, 5, 2, 0 };
```

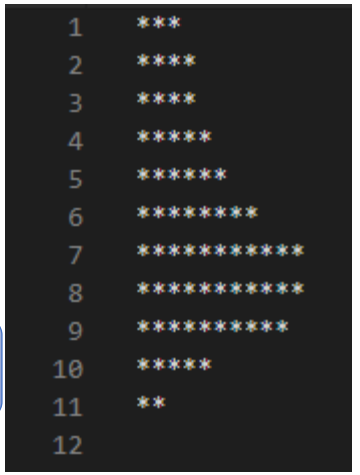
  - A trick how to determine the size of the array

```
int tempsCount = sizeof(temps) / sizeof(temps[0]);
```

    - Funguje jen pro pole se známou délkou

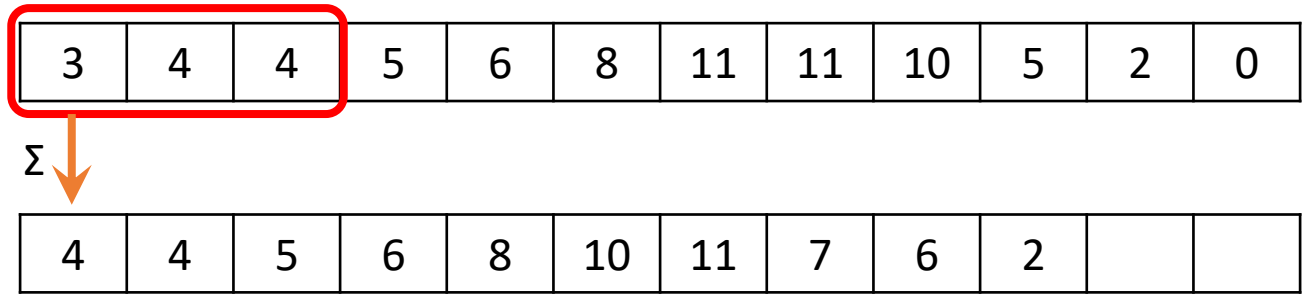
```
int avg(int a[], int count) {
  // sizeof(a) zde nebude fungovat dle očekávání
}
```

Velikost struktury v bytech



Cvičení

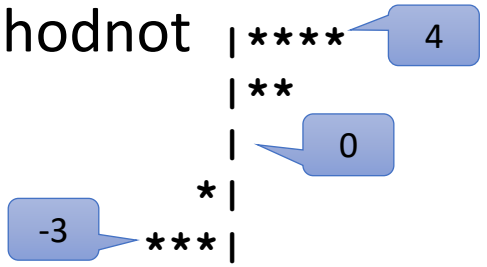
- T7: Napište funkci která spočítá klouzavý průměr (moving average) hodnot
  - Velikost okna dostanete jako parametr
  - Výsledné průměry ukládejte do nového pole
    - Vytvořte si nové pole stejné velikosti jako to vstupní (některé prvky zůstanou nevyužité)
  - Nakonec použijte funkci z T6 pro vypsání výsledného pole



Domácí úloha

Úloha kterou je nutno odevzdat do ReCodExu

- Vertikální graf který umožňuje i záporné hodnoty
  - Podobné jako T6, ale nutnost speciálního ošetření záporných hodnot
  - Hodnota 0 je reprezentována speciálním znakem |
  - Graf je doplněn mezerami dle potřeby (nutno zarovnat |)
  - Hodnoty pochází z nespolehlivého zdroje (mohou chybět)
    - Chybějící hodnoty nahraďte poslední validní hodnotou (nebo 0 na začátku)




```
constexpr int no_value = -999;
int temps[] = { no_value, 10, 9, no_value, 7, no_value, no_value, 1 };
```

```
{ 0, 10, 9, 9, 7, 7, 7, 1 }
```

Závěr

- Vypracovat úlohu v ReCodExu
- Půjčit/koupit si Arduino s funshieldem
- Nainstalovat si Arduino IDE