Programování (PRO-B)

1. cvičení.

Programování v MATLABu

Garant: Ing. Michal MOUČKA, Ph.D.

Vyučující: Ing. Andrij ŠYNKARENKO, Ph.D. andrii.shynkarenko@tul.cz

Ing. Miroslav VAVROUŠEK, Ph.D. miroslav.vavrousek@tul.cz

Katedra výrobních systémů a automatizace (KSA) (ksa.tul.cz)

Určen: pro studenty bez a s malou zkušeností s programováním

Délka: 14 cvičení; 2 písemky; klasifikovaný zápočet

On-line: https://elearning.tul.cz **KSA/PRO-B - Programování (2023)**

Klasifikovaný zápočet

1. Docházka (maximálně **3 absence**);
2. **Dvě** písemky (max 200 bodů);
3. Aktivita během cvičení (+/- 2 bodů/cv);
4. Domácí úkoly (min -20, max 20 bodů);
5. Absolvování online kurzů (max 40 bodů).

Počet bodů	Plnění úkolů
-10	Ani nezkusil(a) vyřešit DU
0	Zkusil(a) ale neúspěšně
5	Vyřešen správně jeden ze dvou příkladů
10	Vyřešené dva ze dvou příkladů

Počet bodů	Název kurzu	Deadline	Odkaz
+10	MATLAB Onramp	Po 1 testu	https://www.mathworks.com/learn/tutorials/matlab-onramp.html
+30	MATLAB Fundamentals (80 %)	Před 2 testem	https://www.mathworks.com/learn/online-courses/matlab-fundamentals.html

Klasifikovaný zápočet. Příklad

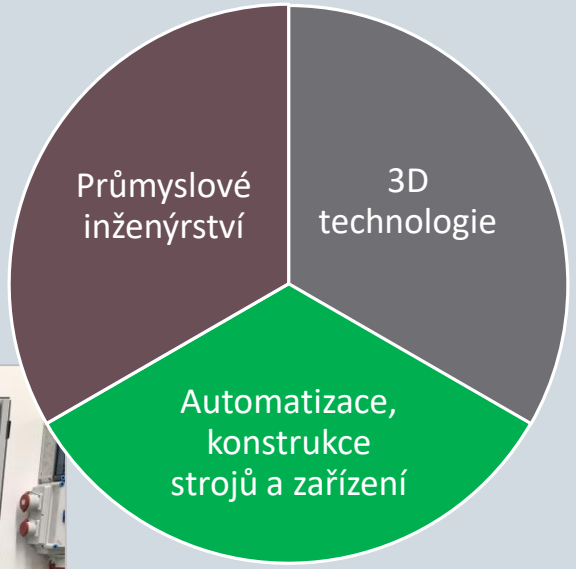
Získané body	Výsledná známka
220 a více	**1 a vynulování počtů absence**
206 a více	**1**
191 – 205	**1-**
176 – 190	**2**
161 – 175	**2-**
146 – 160	**3**
145 a méně	**4 (absolvování kurzu znova)**

Katedra Výrobních systémů a Automatizace

Řešení témat závěrečných prací studentů

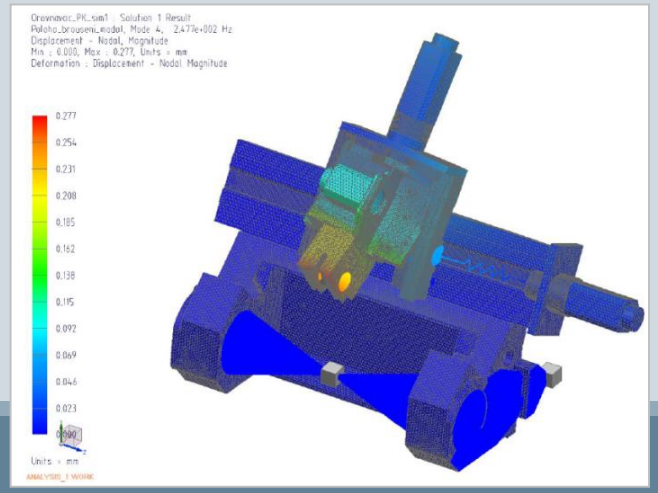
- **Bakalářské práce**
- Diplomové práce

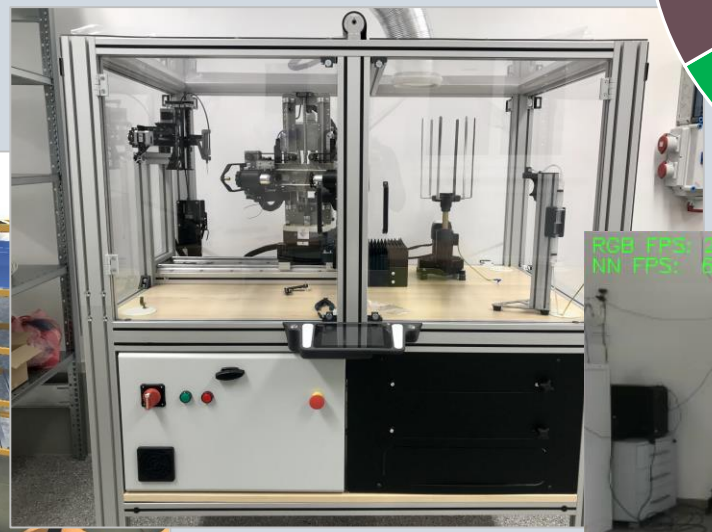
ksa.tul.cz

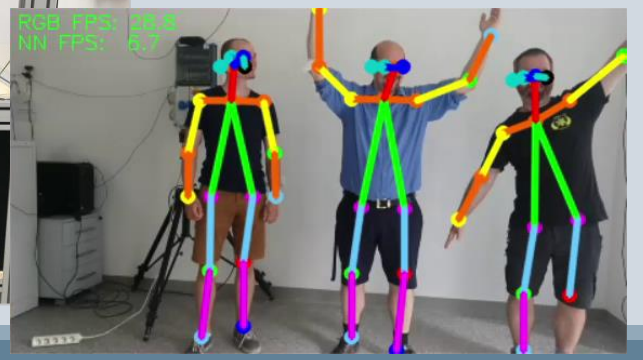


Navazující magisterské studijní programy

- Inovační a průmyslové inženýrství
- Konstrukce strojů a zařízení







Proč programovat?

Zajištění zaměstnání

- Vzhledem k tomu, že technologický růst nezastaví, bude vždy vysoká poptávka po programátorech

Možnost práce z domova

- Vzhledem k tomu, že práce vyžaduje pouze počítač s připojením k internetu, zaměstnanci často mají možnost pracovat na dálku

Zlepšení dovedností řešení problémů a abstrakce

- Když se naučíte programovat, nejde jen o znalosti, které získáte, ale také (a zejména) o užitečné přenositelné dovednosti, které získáte

Kdokoli se může naučit programovat

- Jedna z nejlepších věcí na kódování je, že kdokoli se může naučit, jak to dělat

Proč by měl strojař programovat?

Komplexní systémy

- Mohou navrhovat a vyvíjet komplexní systémy, které vyžadují jak hardware, tak software.

Nové technologie a inovace

- Tato kombinace dovedností může být velmi užitečná při vytváření nových technologií a inovací.

Vývoj nových strojů nebo zařízení

- Strojní inženýr navrhne a vyvine nový stroj nebo zařízení
- Programátor vyvinout software, který by tento stroj nebo zařízení ovládal
- Pochopení obou oblastí vám dá velké uplatnění a role klíčového pracovníka

Co je programování?

Programování je proces od návrhu řešení problému pomocí výpočetní techniky ke spustitelnému počítačovému programu.

Zahrnuje činnosti jako:

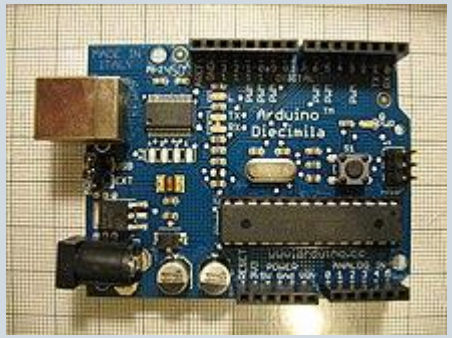
- Analýza problému
- Porozumění problému
- Nalezení algoritmu
- Zápis zdrojového kódu v cílovém programovacím jazyce

Programování si lze představit jako spolupráci mezi lidmi a počítači, při níž lidé vytvářejí instrukce pro počítač (kód) v jazyce, kterému počítače rozumí.

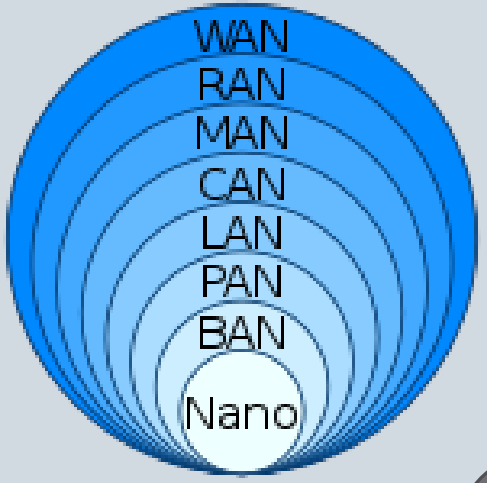
Mnoho světů softwaru

Mobilní platformy

Desktop platformy



Embedded system



Internet

Server platformy

Průmyslové aplikace

Cloudové a webové aplikace

Algoritmus

Jednoznačný a konečný popis postupu, jak vyřešit danou úlohu

Co musí „dobrý“ algoritmus splňovat:

- Elementárnost
  - Složen z jednoduchých (elementárních) kroků.
- Konečnost
  - Konečný počet kroků. Počet kroků může být libovolně velký a lišit se podle rozsahu a hodnot vstupních údajů.
- Obecnost (univerzálnost)
  - Algoritmus neřeší jeden konkrétní problém, ale obecně třídu problémů.
  - Neřeší výpočet pro konkrétní hodnoty (4 na 8), ale řeší se obecně – například mocnění s celočíselným exponentem.
- Determinismus
  - Algoritmus je determinovaný, pokud pro stejné vstupy, poskytuje stejný výstup.
  - Každý krok algoritmu musí být jednoznačně a přesně definován.
  - Ne všechny algoritmy jsou deterministické – obsahují například nějakou náhodnost.
- Výstup
  - Algoritmus má alespoň jeden výstup, veličinu, která je v požadovaném vztahu k zadaným vstupům, a tím tvoří odpověď na problém.

Algoritmus – Jak ho zapsat

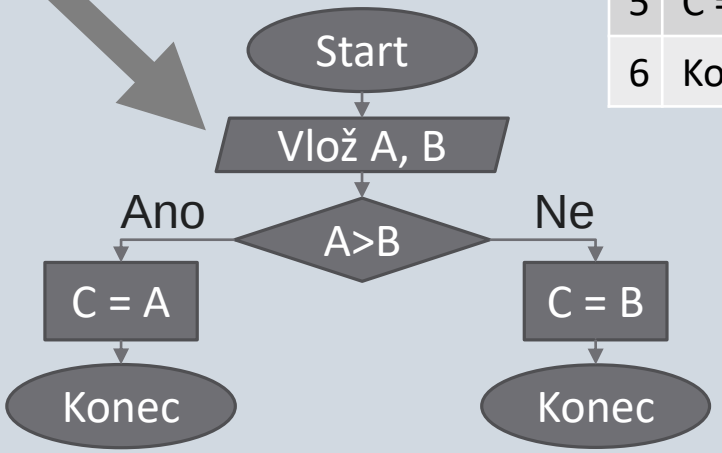
Jednoznačný a konečný popis postupu, jak vyřešit danou úlohu

Způsoby popisu algoritmu

- Slovní popis
  - Pro člověka přirozené, ale často nejednoznačné
- Tabulka kroku
  - Doplňuje jednoznačnost a elementárnost slovního popisu
- Vývojový digram
  - Lepe demontuje posloupnost kroku a celkovou jednoznačnost popisu
- Počítačový program
  - Zápis srozumitelný pro člověka i pro počítač

Uživatel zadá A a B. Pokud je A>B, vlož do C A, jinak B.

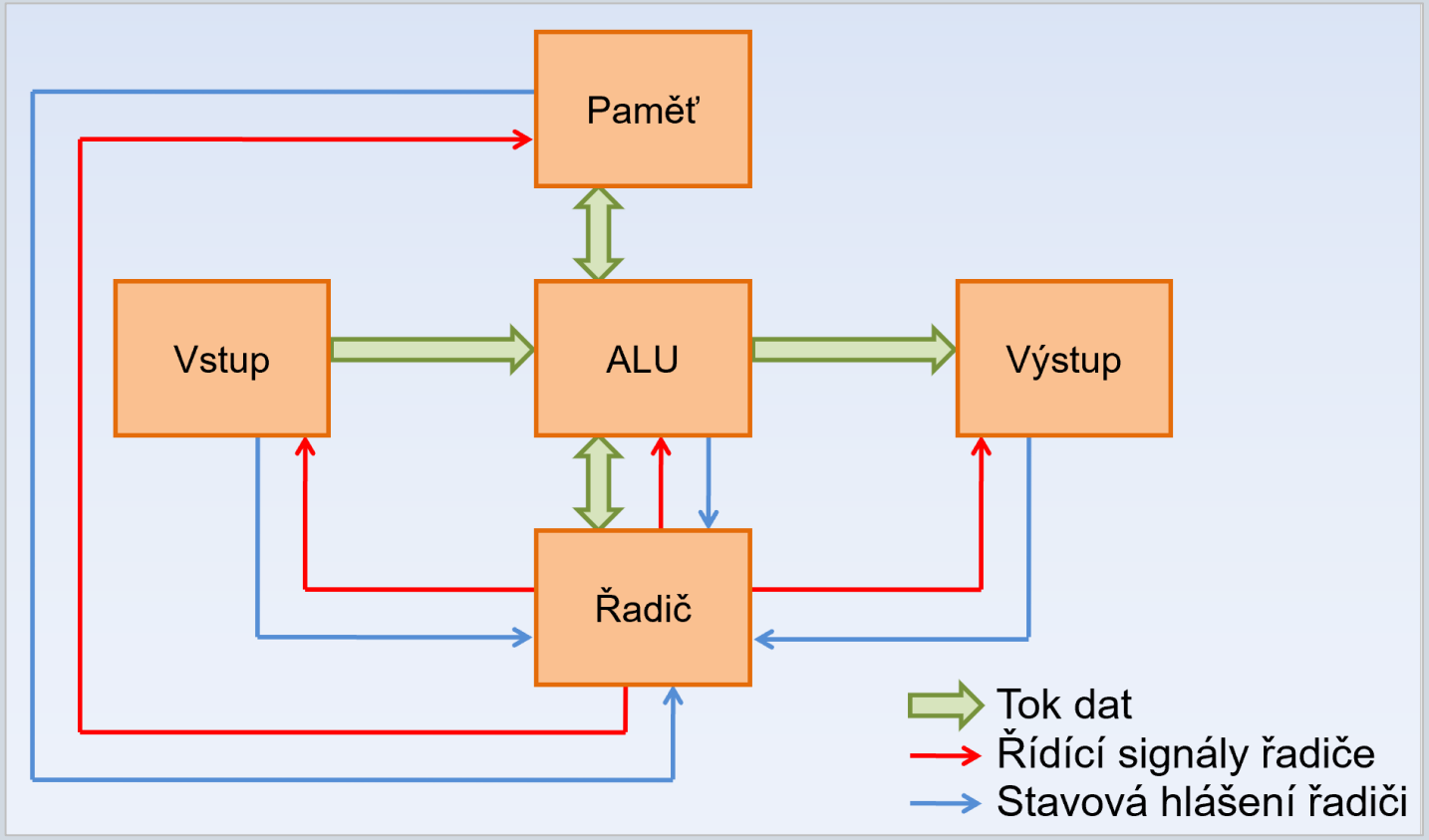
#	Krok
1	Uživatel zadá A a B
2	A>B pokračuj 3, jinak 5
3	C = A
4	Pokračuj 6
5	C = B
6	Konec




```
1 A = input("Zadej A: ")
2 B = input("Zadej B: ")
3
4 if A>B
5     C = A
6 else
7     C = B
8 end
```

Počítače a zpracovaní programu

Von Neumannova architektura počítače



Strojový kód programu

```
movl $0xFF22, %eax
addl %ecx, %edx
xorl %esi, %esi
pushl %ebx
movl 4(%esp), %ebx
leal (%eax, %ecx, 2), %esi
cmpl %eax, %ebx
jnae foo
retl
```

Obsah paměti počítače

```
10111000 00100010 11111111
00000001 11001010
00110001 11110110
01010011
10001011 01011100 00100100 00000100
10001101 00110100 01001000
00111001 11000011
01110010 11101011
11001110
```

Programovací jazyk

Prostředek pro zápis algoritmů, jež mohou být provedeny na počítači.

Nizkoúrovňové jazyky VS Vysokoúrovňové jazyky:

- Nizkoúrovňové – program hardwarově závislý, obtížný zápisu algoritmů
- Vysokoúrovňové – pokročilejší prostředky usnadňující zápis algoritmu

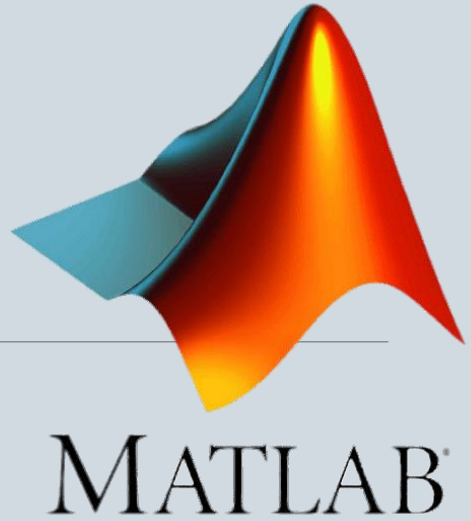
Kompilované jazyky VS Interpretované jazyky

- Kompilované jazyky – zdrojový kód se nejprve přeložit pomocí překladače do strojového kódu
- Interpretované jazyky – zdrojový kód se interpretuje za běhu programu do strojového kódu

Funkcionální jazyky VS Procedurální jazyky

- Procedurální jazyky – strukturované kroky a funkce v programovém kontextu
- Funkcionální jazyky – báze dat, řešitel (solver), dotazy

MATLAB by MathWorks



- Vysokoúrovňový programovací jazyk
- MATLAB je zkratka Matrix Laboratory
- Nabízí interaktivní prostředí s mnoha vestavěnými funkcemi pro technické výpočty, grafiku a animace
- Primárně založen na maticích a polích
- Nabízí široké možnosti vizualizace dat a získávání vhledů do dat pomocí vestavěných grafických funkcí
- MATLAB je vhodný pro experimentování, zkoumání a objevování, ale také k tvorbě uživatelských aplikací

Stáhnout nebo používat on-line verzi:

https://liane.tul.cz/cz/software/MATLAB

Počítačový program

Příkaz – volaní funkce

Příkaz – vstup dat a výpočty

Konstrukce - Podmínka

Zápis algoritmu srozumitelný pro člověka i pro počítač

Program je tvořen:

- Příkazy
  - Provádí se výpočty a operace s daty
  - Příkazy muže být volaní standartních a uživatelských funkcí
- Konstrukce
  - Podmínky a cykly
  - Řídí tok programu
    - Ovlivňují standartní vykonávání programu „řádek po řádku“
- Funkce
  - Samostatná a snadno znovupoužitelná část programu

```
1  clc
2
3  A = input("Zadej A: ")
4  B = input("Zadej B: ")
5
6  if A>B
7      C = A
8  else
9      C = B
10 end
```

Pravidla pro názvy v prostředí MATLAB

Název **proměnné, konstanty, skriptu, funkce** atd.

1. Název je tvořen pouze malými, velkými písmeny a čísly
   - Jsou povoleny některé speciální znaky jako podtržítko (_)
2. Název nesmí začínat číslem ani speciálním znakem
3. Název nesmí být klíčovým slovem
   - Klíčové slovo je vyhrazené slovo v programovacím jazykem se speciálním účelem
   - Slova, která jsou součástí syntaxe MATLABu
     - if, for, while, switch, case, function, end atd.
   - Slova, která jsou součástí jazyků C a C++
     - auto, break, class, const, delete, enum, extern, inline, new, public, return, static, typedef, union atd.

Proměnná a datový typ

Proměnná

- Pojmenované místo paměti, které uchovává hodnotu
- Každá proměnná je určitého datové typu

Datový typ

- definuje druh hodnot, kterých smí nabývat proměnná
  - Celé číslo (Integer)
  - Reálné číslo (Real, Double)
  - Znak (Char)
  - Textu (String)

Skript

```
1  A = input("Zadej A: ")
2  B = input("Zadej B: ")
3
4  if A>B
5      C = A
6  else
7      C = B
8  end
```

Paměť počítače

Adresa	Hodnota	Poznamka
0x00	0x00	A (hodnota 0)
0x01	0xF8	B (hodnota 248)
0x02	0xF8	C (hodnota 248)
0x03	0xAF	
0x04	0xFF	
0x05	0x47	

Konstanta

Konstanta je hodnota, která se nemůže měnit za běhu programu.

Přímá konstanta VS Nepřímá konstanta

- Přímá konstanta – hodnota je přímo zapsaná v kódu
- Nepřímá konstanta – hodnota je pojmenovaná a v kódu se použití pomocí názvu

Každá konstanta je definovaného datového typu

Standardní (z knihoven MATLABu) nebo uživatelem definovaná

```
polomer = 5
obvod = 2*pi*polomer
```

Proměnná

Přímá konstanta

Nepřímá konstanta

Zásady správného programovaní

Obecnost

- kód by měl být napsán tak, aby byl co nejvíce univerzální a mohl být použit v různých situacích

Jednoznačnost

- kód by měl být napsán tak, aby byl snadno pochopitelný a jednoznačný v tom, co dělá

Znuvupoužitelnost

- kód by měl být napsán tak, aby mohl být snadno znovu použit v jiných částech programu nebo v jiných programech

Přehled**nost a čitelnost kódu**

- kód by měl být napsán tak, aby byl snadno čitelný a přehledný pro ostatní programátory
- toho lze dosáhnout například používáním vhodného formátování a komentářů

Úloha

Za jak dlouho dorazí světlo od Slunce na Zemi?

Vzdalenost Země od Slunce: **150 milionů km**

Rychlost světla ve vakuu: **300 000 km/s**