Procedurální programování

Jan Faigl

Katedra počítačů
Fakulta elektrotechnická
České vysoké učení technické v Praze

Přednáška 01

B0B36PRP – Procedurální programování

Přehled témat

- Část 1 – Organizace předmětu
  - Cíle předmětu
  - Prostředky dosažení cílů PRP
  - Hodnocení předmětu a zkouška
- Část 2 – Zadání 1. domácího úkolu (HW00)
- Část 3 – Programování v C
  - První program
  - Příklad použití základní konceptů programování

S. G. Kochan: kapitoly 2, 3

Část I

Organizace předmětu

Předmět a přednášející

B0B36PRP – Procedurální programování

- Webové stránky předmětu
  `https://cw.fel.cvut.cz/wiki/courses/b0b36prp`
- Odevzdávání domácích úkolů
  `https://cw.felk.cvut.cz/upload`
- Přednášející:

  - doc. Ing. **Jan Faigl**, Ph.D.
    - Katedra počítačů – `http://cs.fel.cvut.cz`
    - Centrum umělé inteligence – Artificial Intelligence Center (AIC)
      `http://aic.fel.cvut.cz`
    - Centrum robotiky a autonomních systému
      Center for Robotics and Autonomous Systems – CRAS
      `http://robotics.fel.cvut.cz`
    - Laboratoř výpočetní robotiky (Computational Robotics Laboratory)
      `http://comrob.fel.cvut.cz`

Cíle předmětu

- Osvojit si pohled na výpočetní prostředky jako „*počítačový vědec*“ a naučit se je efektivně používat

  Computer scientist
  - Formulovat problém a jeho řešení počítačovým programem
  - Získat povědomí jaké problémy lze výpočetně řešit
- Získat zkušenost s programováním

  získání vlastní zkušenosti
  - Programování v C

  cvičení, domácí úkoly, zkouška
- Osvojit si schopnost číst, psát a porozumět malým programům
- Získat programovací návyky jak psát
  - srozumitelné a přehledné zdrojové kódy;
  - opakovaně použitelné programy.

Výuka programování

„Separating Programming Sheep from Non-Programming Goats“

http://blog.codinghorror.com/separating-programming-sheep-from-non-programming-goats

http://www.eis.mdx.ac.uk/research/PhDArea/saeed/paper1.pdf

- Efektivní metody výuky programování se hledají již od dob prvních počítačů

  tj. přes více než 50 let
- Přesto se zdá, že je každý základní kurz programování obtížný a 30% až 60% studentů jej na poprvé nezvládne

  V PRP očekáváme průchodnost výrazně vyšší.
- Základní koncept je pochopení principu přiřazení hodnoty proměnné

Test pochopení principu přiřazení

- Zápis programu pro přiřazení hodnot do proměnných *a* a *b* a následné přiřazení proměnné *b* do *a*.

Přiřazení hodnoty proměnné

```
1 int a = 10;
2 int b = 20;
3
4 a = b;
```

- Jaké jsou hodnoty proměnných *a* a *b*?

a. a = 20, b = 0
b. a = 20, b = 20
c. a = 0, b = 10
d. a = 10, b = 10
e. a = 30, b = 20
f. a = 30, b = 0
g. a = 10, b = 30
h. a = 0, b = 30
i. a = 10, b = 20
j. a = 20, b = 10

Uživatelé počítačů

„Uživatel“

- Spouštěč programů
- Zadává vstup

  Píše, kliká, dotýká se
- Čeká na výstup
- Čte výstup
- Relativně omezená množina vstupů

  Pouze to co je dovoleno
- Omezen povrchovou znalostí

  Toho co je mu dovoleno vidět

„Programátor“

- Spouští programy
- Dává počítači příkazy

  Řadí je do posloupnosti
- Vytváří nové programy
- Kombinuje příkazy
- Rozmanitější možnosti použití

  Omezen pouze limity počítače
- **Chápe a rozumí principům**

  Rychle se učí nové technologie!

Způsob reprezentace znalostí

- Z hlediska výpočtu můžeme rozlišit dva základní typy znalostí

Způsoby popisu problému

Deklarativní

- Tvrzení popisující stav
- Axiomatické
- Umožňuje jednoduše ověřovat (testovat) pravdivost tvrzení
- Neposkytuje návod jak hodnotu vypočítat

Příklad:

$$\sqrt{x} = y, y^2 = x, x \geq 0, y \geq 0$$

Imperativní

- Popis jak něco vypočítat
- Posloupnost výpočtu
- Test jak ovlivnit průběh výpočtu

Příklad:

1. `If` $y^2 \approx x$
2. `Then`
   return y
3. `Else`
   $y \leftarrow \frac{y+\frac{x}{y}}{2}$
   `Go to` Step 1

Program je „recept“

- Program je „recept“ – posloupnost kroků (výpočtů) popisující průběh řešení problému
- Programování je schopnost samostatně
  - tvořit programy
  - dekomponovat úlohy na menší celky
  - sestavovat z dílčích částí větší programy řešící komplexní úlohu

B0B36PRP – je příležitostí, jak se těmto schopnostem naučit

Organizace a hodnocení předmětu

- B0B36PRP – Procedurální programování
- Rozsah: 2p+2c; Zakončení: Z,ZK; Kredity: 6;

Z – zápočet, ZK – zkouška

- **Průběžná práce v semestru** – domácí úkoly a test
- **Zkouškový test a implementační zkouška**, případně ústní zkouška

Schopnost samostatné práce na počítačích v učebnách

- Docházka na **cvičení** a odevzdání domácích úloh

Samostatná práce

- Konzultační cvičení (přednáška/seminář)
  - Pátek od 14:45 až 16:00 místnost KN:E-107

- „Alternativní“ absolvování předmětu pro velmi zkušené

Předmět A4B36ACM

Zdroje a literatura

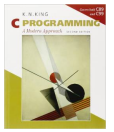
- **Knihy (učebnice)**

„Programming in C“ (Kochan, 2014) nebo „Učebnice jazyka C“ (Herout, 2015)

Programming in C, 4th Edition, *Stephen G. Kochan*, Addison-Wesley, 2014, ISBN 978-0321776419

Základní učební text

C Programming: A Modern Approach, 2nd Edition, *K. N. King*, W. W. Norton & Company, 2008, ISBN 860-1406428577



- Přednášky – podpora učebního textu, slidy, poznámky a především **vlastní poznámky**

Součástí přednášek jsou také zdrojové kódy s příklady!

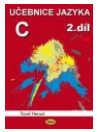
- Cvičení – získání praktických dovedností řešením domácích úkolů a dalších úloh

programovat, programovat, programovat

Další učebnice jazyka C

- The C Programming Language, 2nd Edition (ANSI C), *Brian W. Kernighan, Dennis M. Ritchie*, Prentice Hall, 1988 (1st edition – 1978)

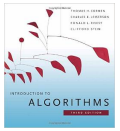
- Učebnice jazyka C – 2. díl, IV. vydání, *Pavel Herout*, KOPP, 2008, ISBN 978-80-7232-367-8



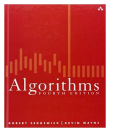
- 21st Century C: C Tips from the New School, *Ben Klemens*, O'Reilly Media, 2012, ISBN 978-1449327149

Další zdroje

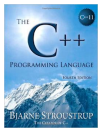
- Introduction to Algorithms, 3rd Edition, *Cormen, Leiserson, Rivest, and Stein*, The MIT Press, 2009, ISBN 978-0262033848



- Algorithms, 4th Edition, *Robert Sedgewick, Kevin Wayne*, Addison-Wesley, 2011, ISBN 978-0321573513



- The C++ Programming Language, 4th Edition (C++11), *Bjarne Stroustrup*, Addison-Wesley, 2013, ISBN 978-0321563842



Přednášky – zimní semestr (ZS) akademického roku 2019/2020

- Harmonogram akademického roku 2019/2020

  http://www.fel.cvut.cz/cz/education/harmonogram1920.html

- Přednášky:
  - Dejvice, místnost T2:D3-309, středa, 16:15–17:45
- 14 výukových týdnů

  14 přednášek
- Konzultační (seminární) přednáška: KN:E-107 (14:45-16:00)
  - Úvod do ovládání počítače a vývojových nástrojů
  - Prezentace "špatných" řešeních
  - Hromadné konzultace dle dotazů a těžkostí

Cvičící

- Ing. **Rudolf Jakub Szadkowski**

- Ing. **Jan Bayer**

- Bc. **Miroslav Tržil**

- Bc. **Jiří Kubík**

- Bc. **Martin Zoula**

- Ing. **Martin Mudroch**, Ph.D.

- RNDr. **Ladislav Serédi**

- Ing. **Petr Váňa**
  BRUTE

- Bc. **Jakub Sláma**

Řešení problémů související s PRP

- Obracejte se na svého cvičícího dle cvičení, na které chodíte (jste přihlášeni)
- Komunikovat můžete elektronickou poštou (e-mail)
  - Pište ze své fakultní adresy (odesílatel)
  - Do předmětu zprávy uvádějte zkratku předmětu PRP
  - Kopii zprávy (Cc) posílejte též příslušnému vedoucímu cvičení (dle studijního programu)
  - V případě zásadních problému (např. týkajících se zápočtu) uvádějte do Cc též přednášejícího

Počítačové laboratoře

- Síťové bootování a síťové domovské adresáře (NFS v4)
- Vývoj v C: *Přenos a synchronizace souborů – ownCloud, SSH, FTP, USB*
  - Překladače **gcc** a **clang** – https://gcc.gnu.org a http://clang.llvm.org
  - Sestavení projektu nástrojem **make** (GNU make)
    Ukážeme si později na přednáškách a cvičení
  - Textový editor – **gedit**, ale také atom, sublime, **vim**, **sublime**
    https://atom.io/, http://www.sublimetext.com/
    http://www.root.cz/clanky/textovy-editor-vim-jako-ide
    Pokud programovat umíte, investuje čas do efektivního ovládání editoru, např. **vim**
  - **Visual Studio Code** (VSC) - https://code.visualstudio.com/
  - C/C++ vývojová prostředí – **WARNING: Do Not Rely on an IDE**
    http://c.learncodethehardway.org/book/ex0.html
    - Geany, Code::Blocks, CodeLite
      https://www.geany.org/, http://www.codeblocks.org, http://codelite.org
    - **CLion**, NetBeans 8.0 (C/C++), Eclipse-CDT – https://www.jetbrains.com/clion
- Odevzdávání domácích úkolů BRUTE https://cw.felk.cvut.cz/upload
  BRUTE – Bundle for Reservation, Uploading, Testing and Evaluation

Služby akademické sítě – FEL, ČVUT

- http://www.fel.cvut.cz/cz/user-info/index.html
- Diskové úložiště ownCloud – https://owncloud.cesnet.cz
- Zasílání velkých souborů – https://filesender.cesnet.cz
- Rozvrh a termíny – FEL Portal – https://portal.fel.cvut.cz
- FEL Google Account – autentizovaný přístup do Google Apps for Education
  Více viz http://google-apps.fel.cvut.cz/
- Gitlab FEL – https://gitlab.fel.cvut.cz/
- Přístup k informačním zdrojům (IEEE Xplore, ACM, Science Direct, Springer Link)
  https://dialog.cvut.cz
- Akademické a kampusové licence
  https://download.cvut.cz
- Národní Gridová Infrastruktura MetaCentrum
  http://www.metacentrum.cz/cs/index.html

Domácí úkoly a další úlohy

- Samostatná práce s cílem osvojit si praktické zkušenosti
- Jednotné zadání na přednášce a jednotný termín odevzdání
- Odevzdání domácích úkolů prostřednictvím Upload system
  https://cw.felk.cvut.cz/upload
  - Nahrání (upload) archivu s nezbytnými zdrojovými soubory
  - Ověření správnosti implementace automatickými testy
  - Penalizace za překročení počtu uploadů
    Odevzdávejte funkční kódy, nikoliv „pouze“ kódy, které projdou testy
  - Detekce plagiátů
    Cílem řešení úkolů je získat vlastní zkušenost
- Úkoly jsou jednoduché a navrhované tak, aby byly stihnutelné
- Klíčem k úspěšnému dokončení předmětu je samostatná práce a osvojení si technik a znalostí
  Průběžná práce a řešení úkolů
- Pokud něčemu nerozumíte, **ptejte se**!
  Pokud chybujete, tak se učíte, pokud nechybujete, tak už to umíte!

Přehled domácích úkolů

- Domácí úkoly s povinným, volitelným, případně bonusovým zadáním
  `https://cw.fel.cvut.cz/wiki/courses/b0b36prp/hw/start`
  0. HW 00 - První program
  1. HW 01 - Načítání vstupu, výpočet a výstup
  2. HW 02 - První cyklus (**Kontrola stylu** – až -100% z dosažených bodů)
  3. HW 03 - Kreslení (ASCII art)
  4. HW 04 - Prvočíselný rozklad (**Kontrola stylu** – až -100% dsž. bodů)
  5. HW 05 - Maticové počty
  6. HW 06 - Caesarova šifra (**Kontrola stylu** – až -100% dsž. bodů)
  7. HW 07 - Hledání textu v souborech
  8. HW 08 - Kruhová fronta v poli
  9. HW 09 - Načítání a ukládání grafu
  10. HW 10 - Integrace načítání grafu a prioritní fronta v úloze hledání nejkratších cest

  HW 09 + 12. přednáška, soutěž na extra body
- Podmínkou zápočtu je úspěšné odevzdání všech domácích úkolů
- Odevzdání volitelného zadání je doporučeno (není částečné odevzdání)

Celkové body za povinné zadání **25b**, volitelné zadání **20b**, bonusové **10b+**

Odevzdávání domácích úkolů – BRUTE

- **BRUTE** – Bundle for Reservation, Uploading, Testing and Evaluation
  - Formální kontrola – kompilace programu
  - Testování funkčnost a správnosti – kontrola výstupu pro daný vstup
    - Veřejné vstupy a odpovídající výstupy / neveřejné vstupy
  - Před uploadem programu si program otestujete sami
    - S využitím dostupných vstupů a výstupů
    - Vytvoření vlastních vstupů a laděním programu
    - Vytvoření vstupů přiloženým generátorem vstupů
    - Ověření výstupu přiloženým testovacím nebo referenčním programem
- Porozumění kódu a kontrola možných stavů
  - Pro každý řádek byste měli být schopni odpovědět proč tam je a co dělá
  - Pro kažou funkci nebo načtení vstupu od uživatele analyzujte možné vstupní hodnoty nebo návratové hodnoty funkcí
    - Pokud je z hlediska funkčnosti vstup nebo návratová hodnota zásadní, proveďte kontrolu vstupu a/nebo příslušnou akci, např. vypsání hlášení a ukončení programu

Např. očekávaný vstup je číslo a uživatel zadá něco jiného.

Úkoly a BRUTE

- Úkoly nejsou jen o odevzdání implementace, která projde BRUTE testy
- Úkoly jsem především o postupném získání zkušeností s konkrétními
- Úkoly mají relativní obtížnost velmi podobnou
  - Je důležité postupně samostatně řešit jednotlivé úkoly a osvojovat si dílčích dovednosti

Absolutně jsou úlohy postupně náročnější

- Netrapte se s řešením příliš dlouho sami, ptejte se na fóru, cvičících, přednášce, **konzultačním semináři**
- Úlohy HW02, HW04 a HW06 budou manuálně kontrolovány na kvalitu kódovacího stylu
  - Zaměřeno na konzistenci, čitelnost, a **modularitu** (rozdělní do funkcí)
  - *Také proto, že studenti tráví mnoho času implementací aniž by měli v implementaci nějaký výrazný progress.*

Hodnocení předmětu

Zdroj bodů	Maximum bodů	Přípustné minimum bodů
Domácí úkoly	45	-
Bonusové úkoly	10^+	-
Test v semestru	5	-
Písemný zkouškový test	20	10
Implementační zkouška	20	**10**
Ústní zkouška	-	-
Součet	100^+ bodů	

- **Zápočet**: nejméně 30 bodů ze semestru a odevzdáné všechny domácí úkoly a to **nejpozději do 10.1.2020 ve 23:59 CET!**
- Předmět lze úspěšné ukončit **zápočtem** a **zkouškou**
- Test a písemná zkouška – krátké stručné odpovědi prokazující porozumění problematice `https://cw.fel.cvut.cz/b191/courses/b0b36prp/resources/test`
- Implementační zkouška – prokázaní samostatně porozumět a napsat krátký program – `https://cw.fel.cvut.cz/b191/courses/b0b36prp/resources/exam`

Klasifikace předmětu

Klasifikace	Bodové rozmezí	Hodnocení	Slovní hodnocení
A	≥ 90	1	výborně
B	80–89	1,5	velmi dobře
C	70–79	2	dobře
D	60–69	2,5	uspokojivě
E	50–59	3	dostatečně
F	<50	4	nedostatečně

- Včasné odevzdáním všech domácích úkolů s povinným a **volitelným** zadáním (45 bodů)
- Bonusová úloha a bonusové odevzdání HW10 (5 bodů)
- Test v semestru (5 bodů)
- Písemná zkouška (20 bodů) 15 a více bodů je velmi slušný výsledek
- Implementační zkouška (20 bodů)
- **95 bodů** a více (A – výborně), **76 bodů** (C – dobře) – (20% ztrata)

- Body jsou indikátorem průběžných výsledků

Zkouška může známku zlepšit, ale také v případě zásadní neznalosti zhoršit

Přehled přednášek

- 01 - Informace o předmětu, Procedurální programování
- 02 - Základy programování v C *S. G. Kochan: kapitoly 2 a 3*
- 03 - Zápis programu v C a základní řídicí struktury *S. G. Kochan: kapitoly 3, 4, 5 a část 6*
- 04 - Řídicí struktury, výrazy a funkce *S. G. Kochan: kapitoly 4, 5, 6 a 12*
- 05 - Pole, ukazatel, textový řetězec, vstup a výstup programu *S. G. Kochan: kapitoly 7, 10 a 11*
- 06 - Ukazatele, paměťové třídy a volání funkcí *S. G. Kochan: kapitoly 8 a 11*
- 07 - Struktury a uniony, přesnost výpočtů a vnitřní reprezentace číselných typů *S. G. Kochan: kapitoly 9, 14, 17 a Appendix B*
- 08 - Standardní knihovny C. Rekurze. (Základní vlastnosti jazyka C probrány.)
- 09 - Spojové struktury *S. G. Kochan: kapitola 16 a Appendix B*
- 10 - Stromy
- 11 - Abstraktní datový typ (ADT) - zásobník, fronta, prioritní fronta
- 12 - Prioritní fronta, halda. Příklad použití při hledání nejkratší cesty v grafu
- 13 - Rezerva – např. diskuse, dotazy, nebo *Systémy pro správu verzí* nebo *C vs C++*
- 14 - Zkouškový **TEST**

Přednáška nemusí být prezentace slidů – je očekávána interakce, řešení dotazů a diskuse problematický a náročnějších částí

Podklady k přednášce jsou k dispozici před přednáškou podobně jako učebnice

Část II

Část 2 – Zadání 0. domácího úkolu (HW00)

Zadání 0. domácího úkolu HW00

Téma: **První program**

Povinné zadání: **1b**; Volitelné zadání: není; Bonusové zadání: není

- **Motivace**: Seznámení se s odevzdávacím systémem BRUTE
- **Cíl**: Osvojit si kompilaci a odevzdávání domácích úkolů
- **Zadání**: `https://cw.fel.cvut.cz/wiki/courses/b0b36prp/hw/hw00`
  - Napište program, který vytiskne na obrazovku text `Hello PRP!` zakončený znakem nového řádku `\n`
- Termín odevzdání: 28.09.2019, 23:59:59 PDT

PDT – Pacific Daylight Time

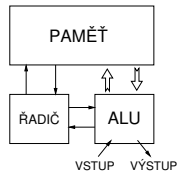
Část III

Část 3 – Programování v C

Počítač „počítá“, tedy pracuje s čísly

- Výpočet je realizován *aritmeticko-logickou jednotkou* (ALU)
- Číselné hodnoty jsou uloženy v paměti počítače
- Předpis jak a co počítat je zapsán programem

  Opět jako posloupnost číselných hodnot se specifickým významem



- Základní jednotkou uložení informace v paměti počítače je bit (binární hodnota 0 nebo 1)

  Historicky vychází z děrného štítku - zápis a strojové zpracování informací
- ALU pracuje s vyhrazenou pamětí
  např. součet dvou hodnot 10 + 4 může být realizován

registry	akumulátorem
`mov $10, %r1`	`lda $10`
`mov $04, %r2`	`add $04`
`add r1, r2, r3`	`sto r3`

Každá instrukce má svůj příslušný zápis jako číselná hodnota (opcode), program je tak posloupnost číselných hodnot

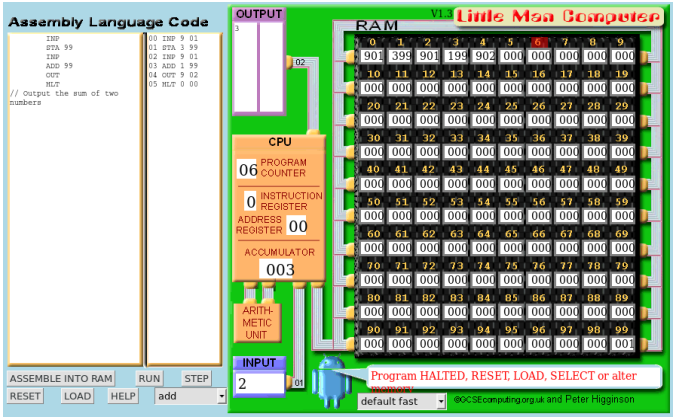
Princip výpočtu

- Pochopení principu výpočtu může pomoci simulátor procesoru např. Little Man Computer

  https://peterhigginson.co.uk/LMC/, https://gcsecomputing.org.uk/lmc/

  http://www.vivaxsolutions.com/web/lmc.aspx

  https://www.youtube.com/watch?v=6cbJWV4AGmk



Zápis programu

- Zápis instrukcí v „*opkódech*“ je možný, ale není příliš pohodlný
  - Číselné hodnoty jsou použity pro identifikaci operací a také míst v paměti, na kterých jsou uložena data (opět jako číselné hodnoty)
- Textový zápis pojmenovaných instrukcí procesoru (assembler) může být srozumitelný, ale je relativně dlouhý
- Přehlednost zápisu a schopnost orientovat se v kódu je jednou z motivací vzniku různých programovacích jazyků
- Jedním z jazyků nabízející kompromis mezi srozumitelností, čitelností a efektivitou zápisu je jazyk C

Programu v Cčku

- Pamětová místa s daty jsou „odkazována“ proměnnými
  - Typ proměnné definuje kolik paměti je použito pro uložení dat (číselné) hodnoty
  - Např. zavedení proměnných pro uložení celých čísel typu `int`

```
int a;
int b;
int c;
```

- Dále používáme obvyklý zápis operací

```
a = 10;
b = 4;
c = a + b;
```

Zápis uloží hodnotu 10 na paměťové místo odkazované proměnnou `a`, hodnotu 4 na paměťové místo odkazované proměnnou `b` a následně provede součet hodnot, který uloží na pamětové místo odkazované proměnnou `c`.

Základní koncepty programování

V programování jsou využívány tři klíčové koncepty, kterou jsou vzájemně kombinovány a umožňují vytvářet komplexní programy.

- **Přiřazení** - uložení hodnoty na definované místo v paměti
- **Větvení** - volba posloupnosti instrukcí na základě hodnoty nějaké proměnné (místa v paměti)
- **Cyklus** - Opakování nějaké posloupnosti instrukcí s novými daty

Abychom mohli lépe a snadněji organizovat posloupnosti instrukcí do složitější celků, je vhodné program strukturovat do znovupoužitelných částí: **procedur** a **funkcí**

- Procedura představuje předpis co se má s jednolivými paměťovými místy provádět
- Výsledek procedury závisí na hodnotách uložených v paměti
- Procedura/funkce/algoritmus řeší obecnou úlohu nějakého výpočtu

Neméně důležitým konceptem je tak zobecňování výpočtu, které vlastně „zjednodušuje“ řešení problémů.

Program v C

- Program v C je organizován do funkcí
- U spustitelného programu musíme označit, která funkce se má spustit jako první
- V Cčku je to funkce pojmenovaná `main()`

```
1  void main(void)
2  {
3      int a;
4      int b;
5      int c;
6
7      a = 10;
8      b = 4;
9      c = a + b;
10 }
```

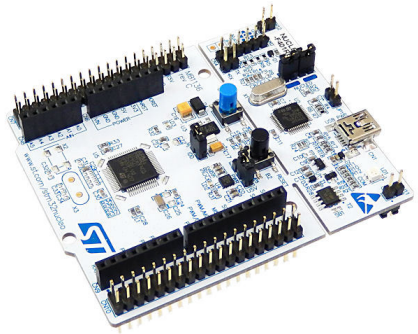
lec01/add.c

Program můžeme zkompilovat a spustit, ale úplně nemáme přehled co dělá a jaký je výsledek. Program přímo neinteraguje s uživatelem.

Interaktivní program

- Proměnné (paměťová místa) mohou přímo reprezentovat periferie - např. tlačítko (0 - stisknuto) a LED (1 - svítí).
- Ovládání LED tlačítkem tak můžeme realizovat jako nekonečnou smyčku, ve které nastavujeme hodnotu LED podle stisknutého nebo nestiknutého tlačítka

```
1  #include "mbed.h"
2
3  DigitalOut myled(LED1);
4  DigitalIn mybutton(USER_BUTTON);
5
6  int main()
7  {
8     while (1) {
9        if (mybutton == 0) {
10          myled = 1;
11       } else {
12          myled = 0;
13       }
14    }
15 }
```



Textově orientovaná interakce s uživatelem

- Dalším ze způsobů interakce s uživatelem je textový výstup a vstup.
- V případě programu běžícího v rámci operačního systému je nutné využívat služby operačního systému realizující interakci s uživatelem
- Proto musíme v Cčkovém programu přidat podporu pro vstup a výstup, např. knihovnu `stdio.h`

```c
#include <stdio.h>

int main(void)
{
    printf("I like B0B36PRP!\n");

    return 0;
}
```

lec01/program.c

Program zároveň vrací návratovou hodnotu a tím komunikuje s uživatelem nebo nadřazeným programem, který tak může identifikovat jakým způsobem byl program ukončen.

Příklad opakovaného tisku na základě uživatelského vstupu

- Úkol: Uživatel zadá počet opakování tisku zprávy a pokud je počet větší než 0 a zároveň menší než 10 vypíše zprávu tolikrát kolik bylo zadáno. V opačném případě upozorní uživatele na omezený rozsah.
  - **Přirazení** - uložení hodnoty počtu opakování od uživatele (proměnná `n`)
  - **Větvení** - kontrola mezí vstupní hodnoty
  - **Cyklus** - opakování vypisu *n* krát
    - Při opakovaném průchodu cyklem počítáme kolikrát byla zpráva vytištěna (řídicí proměnná `i`)
  - Načtení vstupu od uživatele realizujeme funkcí `scanf()`

Popis příště a vysvětlení syntaxe v dalších přednáškách

Příklad řešení

```c
#include <stdio.h>

int main(void)
{
   int ret = 0;
   int n;
   printf("Enter a positive integer number from 1 to 9: ");
   scanf("%d", &n);
   if (n > 0 && n < 10) {
      int i = 0;
      while (i < n) {
         printf("I like B0B36PRP!\n");
         i = i + 1;
      }
   } else {
      printf("Input value must be in the range (0,10)\n");
      ret = -1;
   }
   return ret;
}
```

lec01/print.c

Shrnutí přednášky

Diskutovaná témata

- Informace o předmětu
- Procedurální programování (v C)

- Příště: Základy programování v C