

Základy programování v C

Jan Faigl

Katedra počítačů
Fakulta elektrotechnická
České vysoké učení technické v Praze

Přednáška 01

B0B36PRP – Procedurální programování

Část I

Organizace předmětu

Přehled témat

- Část 1 – Organizace předmětu
 - Cíle předmětu
 - Prostředky dosažení cílů PRP
 - Hodnocení předmětu a zkouška
- Část 2 – Základy programování v C
 - Program v C
 - Proměnné a jejich hodnoty
 - Základní číselné typy
 - Výrazy a operátory
 - Formátovaný vstup a výstup

S. G. Kochan: kapitoly 2, 3

- Část 3 – Zadání 1. domácího úkolu (HW01)

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 systémů
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.

Test pochopení principu přiřazení

- Zápis programu pro přiřazení hodnot do proměnných a a b a následně 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. $a = 20, b = 0$ | f. $a = 30, b = 0$ |
| b. $a = 20, b = 20$ | g. $a = 10, b = 30$ |
| c. $a = 0, b = 10$ | h. $a = 0, b = 30$ |
| d. $a = 10, b = 10$ | i. $a = 10, b = 20$ |
| e. $a = 30, b = 20$ | j. $a = 20, b = 10$ |

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

Uživatelé počítačů

- | | |
|---|--|
| <h3>„Uživatel“</h3> <ul style="list-style-type: none"> ■ Spouštěč programů ■ Zadává vstup <i>Píše, kliká, dotýká se</i> ■ Čeká na výstup ■ Čte výstup ■ Relativně omezená množina vstupů <i>Pouze to co je dovoleno</i> ■ Omezen povrchovou znalostí <i>Toho co je mu dovoleno vidět</i> | <h3>„Programátor“</h3> <ul style="list-style-type: none"> ■ Spouští programy ■ Dává počítači příkazy <i>Řadí je do posloupnosti</i> ■ Vytváří nové programy ■ Kombinuje příkazy ■ Rozmanitější možnosti použití <i>Omezen pouze limity počítače</i> ■ 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

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
- Implementační a případně ústní zkouška

Schopnost samostatné práce na počítačích v učebnách

- Docházka na cvičení a odevzdání domácích úloh

Samostatná práce

- **Supervize práce v počítačové učebně**

- Pátek od 11:00 až 14:15 místnost T2:H1-131 (7.10.-11.11.2016)

Pro osvojení si základních návyků používání počítačů v učebně a řešení programovacích úloh

- „Alternativní“ absolvování předmětu pro velmi zkušené

Předmět A4B36ACM

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

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




Učebnice jazyka C, VI. vydání, Pavel Herout,
KOPP, 2010, ISBN 978-80-7232-406-4




- 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)



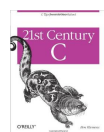
 **Učebnice jazyka C – 2. díl, IV. vydání**, *Pavel Herout*, KOPP, 2008, ISBN 978-80-7232-367-8



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



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



Přednášky – zimní semestr (ZS) akademického roku 2016/2017

■ Harmonogram akademického roku 2016/2017

<http://www.fel.cvut.cz/cz/education/harmonogram1617.html>

■ Přednášky:


- Dejvice, místnost T2:D2-256, úterý, 11:00–12:30
- Dejvice, místnost T2:D3-309, středa, 16:15–17:45

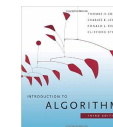
■ 14 výukových týdnů

13 přednášek

■ Středa 16.11.2016 výuka jako pátek

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



Cvičící

■ Ing. **Petr Váňa**
Vedoucí cvičení programu
Otevřená informatika (OI)



■ Ing. **Martin Mudroch, Ph.D.**
Vedoucí cvičení programu
Elektronika a komunikace (EK)



■ Ing. **Petr Čížek**



■ Ing. **Stanislav Vítek, Ph.D.**
Vedoucí cvičení programu
Elektrotechnika, energetika a
management (EEM)



■ Ing. **Daniel Fišer**



■ Ing. **Martin Bloch, CSc.**



■ Ing. **Karel Jalovec**



■ Ing. **Ondřej Nentvich**



■ Ing. **Martin Svatoš**



Řešení problémů související s PRP

- Obracejte se na svého cvičícího dle cvičení, na které chodíte (jste přihlášení)
- 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 problémů (např. týkajících se zápočtu) uvádějte do Cc též přednášejícího

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>

Počítačové laboratoře

- Síťové bootování a síťové domovské adresáře (NFS v4)
Přenos a synchronizace souborů – ownCloud, SSH, FTP, USB
- Vývoj v C:
 - 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, atom, sublime, **vim**
<https://atom.io/>, <http://www.sublimetext.com/>
<http://www.root.cz/clanky/textovy-editor-vim-jako-ide>
 - C/C++ vývojová prostředí – **WARNING: Do Not Use An IDE**
<http://c.learncodethehardway.org/book/ex0.html>
 - Code::Blocks, CodeLite
<http://www.codeblocks.org>, <http://codelite.org>
 - NetBeans 8.0 (C/C++), Eclipse-CDT
 - CLion – <https://www.jetbrains.com/clion>
- Odevzdávání domácích úkolů – Upload system
<https://cw.felk.cvut.cz/upload>

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>

- HW 00 - První program *Testovací úkol za 1 bod*
- HW 01 - Načítání vstupu, výpočet a výstup
- HW 02 - První cyklus
- HW 03 - Kreslení (ASCII art)
- HW 04 - Prvočíselný rozklad
- HW 05 - Maticové počty
- HW 06 - Caesarova šifra
- HW 07 - Lámání hesla
- HW 08 - Kruhová fronta v poli
- HW 09 - Načítání a ukládání grafu
- HW 10 - Integrace načítání grafu a prioritní fronta v úloze hledání nejkratších cest *HW 09 + 11. 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 **35b**, volitelné zadání **25b**, bonusové **10b+**

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

- Očekávané výsledky

- Včasné odevzdáním všech domácích úkolů s povinným a volitelným zadáním (**60 bodů**)
- Test v semestru (**10 bodů**)
- Písemná zkouška (**20 bodů**)
- Implementační zkouška (**10 bodů**)
- 95 bodů** a více (A – výborně) – s menším zaváháním
- 76 bodů** (C – dobře) – při 20% ztrátě

76 a více bodů představuje solidní základ pro budování programátorských dovedností.

Hodnocení předmětu

Zdroj bodů	Maximum bodů (povinné + volitelné)	Přípustné minimum bodů
Domácí úkoly	60	30
Test v semestru	10	0
Písemný zkuškový test	20	10
Implementační zkouška	10	0
Součet	100 bodů	40 bodů je F!

- Pro zápočet je minimální počet bodů ze semestru **30**
- Předmět lze úspěšně ukončit **zápočtem a zkouškou**
- Získání **zápočtu** je podmíněno odevzdáním všech domácích úkolů
Nejpozději 15.1.2017 ve 23:59 CET!

Přehled přednášek

- 01 - Informace o předmětu, Základy programování v C
S. G. Kochan: kapitoly 2 a 3
- 02 - Zápis programu v C a základní řídicí struktury
S. G. Kochan: kapitoly 3, 4, 5 a část 6
- 03 - Řídicí struktury, výrazy a funkce
S. G. Kochan: kapitoly 4, 5, 6 a 12
- 04 - Pole, ukazatel, textový řetězec, vstup a výstup programu
S. G. Kochan: kapitoly 7, 10 a 11
- 05 - Ukazatele, paměťové třídy a volání funkcí
S. G. Kochan: kapitoly 8 a 11
- 06 - Struktury a uniony, přesnost výpočtů a vnitřní reprezentace číselných typů
S. G. Kochan: kapitoly 9, 14, 17 a Appendix B
- x67 - Přednáška na vyzvané téma (7. výukový týden, pouze úterý)
- 07 - Standardní knihovny C. Rekurze. (**Základní vlastnosti jazyka C probrány.**)
S. G. Kochan: kapitola 16 a Appendix B
- 08 - Spojové struktury
- 09 - Stromy
- 10 - Abstraktní datový typ (ADT) - zásobník, fronta, prioritní fronta
- 11 - Prioritní fronta, halda. Příklad použití při hledání nejkratší cesty v grafu
- 12 - *Systémy pro správu verzí*
- 13 - *Rozdíly C a C++*

Část II

Část 2 – Základy programování v C

Zápis programu

- Zdrojový kód programu v jazyce C se zapisuje do textových souborů
 - Zdrojové soubory zpravidla pojmenované s koncovkou `.c`
 - Hlavičkové soubory s koncovkou `.h`
- Kompilací zdrojových souborů překladačem do binární podoby vznikají objektové soubory `.o`
- Z objektových souborů se sestavuje výsledný program
- Příklad zápisu jednoduchého programu:

```

1 #include <stdio.h>
2
3 int main(void)
4 {
5     printf("I like BOB36PRP!\n");
6
7     return 0;
8 }
```

lec01/program.c

Jazyk C

- Nízko-úrovňový programovací jazyk
- Systémový programovací jazyk (operační systém)
 - Jazyk pro vestavné (embedded) systémy — MCU, křížová (cross) kompilace*
- Téměř vše nechává na uživateli (programátorovi)
 - Inicializace proměnných, uvolňování dynamické paměti*
- Má blízko k využití hardwarových zdrojů výpočetního systému
 - Přímé volání služeb OS, přímý zápis do registrů a portů.*
- Klíčové pro správné fungování programu je zacházení s pamětí

Cílem kurzu PRP je naučit se základním principům, které lze následně generalizovat též pro jiné programovací jazyky. Pochopení těchto principů je klíčem k efektivnímu psaní efektivních programů.

Je výhodné mít překlad programu plně pod kontrolou.

Přestože to může z počátku vypadat složitě, jsou základní principy relativně jednoduché. I proto je výhodné používat základní nástroje pro překlad programů a po jejich osvojení využít komplexnější vývojové prostředí.

Překlad (kompilace) a spuštění programu

- Zdrojový soubor `program.c` přeložíme do spustitelné podoby kompilátorem např. `clang` nebo `gcc`
 - `clang program.c`
- Vznikne soubor `a.out`, který můžeme spustit např.
 - `./a.out`
 - Alternativně pouze jako `a.out` pokud je aktuální pracovní adresář nastaven v prohledávané cestě spustitelných souborů*
- Program po spuštění vypíše text uvedený jako argument `printf()`
 - `./a.out`
 - I like BOB36PRP!
- Pokud nechce psát `./a.out` ale raději jen `a.out` lze přidat aktuální pracovní adresář do cest(y) definované proměnnou prostředí `PATH`
 - `export PATH="$PATH: 'pwd' "`
 - Pracovních adresářů můžete mít více—používejte obezřetně.*
- Příkaz `pwd` vytiskne aktuální pracovní adresář, více viz `man pwd`

Ano jde to, ale není dobrý nápad to používat!

Struktura zdrojového souboru

- Komentovaný zdrojový soubor program.c

```

1 /* komentar zapisujeme do dvojice vyhrazenych znaku */
2 // Nebo v C11 jako jednoradkovy
3 #include <stdio.h> /* vlozeni hlavickoveho souboru
   standardni knihovny stdio.h */
4
5 int main(void) // zjednodusena deklarace
6 { // hlavni funkce program main()
7     printf("I like BOB36PRP!\n"); /* volani funkce
   printf() z knihovny stdio.h pro tisk textoveho
   retezce na standardni vystup. Znak \n definuje novy
   radek (odradkovani). */
8
9     return 0; /* ukonceni funkce a predani navratove
   hodnoty 0 operacnimu systemu */
10 }
```

Překlad a sestavení programu

- Uvedený příklad slučuje jednotlivé kroky překladu a sestavení programu do volání jediného příkazu (clang nebo gcc). Překlad se však skládá ze tří částí, které lze provést individuálně

1. Textové předzpracování **preprocesorem**, který má vlastní makro jazyk (příkazy uvozeny znakem #)

Všechny odkazované hlavičkové soubory se vloží do jediného zdrojového souboru

2. Vlastní překlad zdrojového souboru do objektového souboru

Zpravidla jsou jména souborů zakončena příponou .o

clang -c program.c -o program.o

Příkaz kombinuje volání preprocesoru a kompilátoru.

3. Spustitelný soubor se sestaví z příslušných dílčích objektových souborů a odkazovaných knihoven, tzv. „linkováním“ (**linker**), např.

clang program.o -o program

Zdrojové soubory

Proč psát do dvou nebo více souborů?

- Rozdělení na zdrojové a hlavičkové soubory umožňuje rozlišit deklaraci a definici, především však podporuje

- **Organizaci** zdrojových kódů v adresářové struktuře souborů
- **Modularitu**

- Hlavičkový soubor obsahuje popis co modul nabízí
- Popis (seznam) funkcí a jejich parametrů bez konkrétní implementace

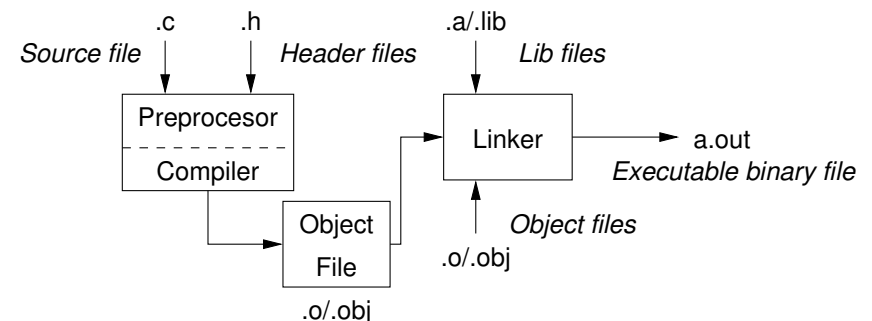
- **Znovupoužitelnost**

- Pro využití binární knihovny potřebuje znát její „rozhraní“, které je deklarované v hlavičkovém souboru

Zatím nemusí být výhody zřejmé, ale budou. V úloze HW 10!

Schema překladu a sestavení programu

- Vývoj programu se skládá z editace zdrojových souborů (.c a .h);
Lidsky čitelných
- kompilace dílčích zdrojových souborů (.c) do objektových souborů (.o nebo .obj) ;
Strojově čitelných
- linkování přeložených souborů do spustitelného programu;
- spouštění a ladění aplikace a opětovné editace zdrojových souborů.



Části překladače a sestavení programu

- **preprocesor** – umožňuje definovat makra a tím přizpůsobit překlad aplikace kompilačnímu prostředí
Výstupem je textový („zdrojový“) soubor.
- **compiler** – Překládá zdrojový (textový) soubor do strojově čitelné (a spustitelné) podoby
Nativní (strojový) kód platformy, bytecode, případně assembler
- **linker** – sestavuje program z objektových souborů do podoby výsledné aplikace
Stále může odkazovat na knihovní funkce (dynamické knihovny linkované při spuštění programu), může též obsahovat volání OS (knihovny).
- Dílčí části **preprocesor**, **compiler**, **linker** jsou zpravidla „jediný“ program, který se volá s příslušnými parametry

Příklad součtu dvou hodnot

```

1 #include <stdio.h>
2
3 int main(void)
4 {
5     int sum; /* definice lokální proměnné typu int */
6
7     sum = 100 + 43; /* hodnota výrazu se uloží do sum */
8     printf("The sum of 100 and 43 is %i\n", sum);
9     /* %i formátovací příkaz pro tisk celého čísla */
10    return 0;
11 }

```

- Proměnná `sum` typu `int` reprezentuje celé číslo, jehož hodnota je uložena v paměti
- `sum` je námi zvolené symbolické jméno místa v paměti, kde je uložena celočíselná hodnota (typu `int`)

Překladače jazyka C

- V rámci předmětu PRP budeme používat především překladače z rodiny:

- **gcc** – GNU Compiler Collection

<https://gcc.gnu.org>

- **clang** – C language family frontend for LLVM

<http://clang.llvm.org>

Pro win* platformy pak odvozená prostředí [cygwin](https://www.cygwin.com/) <https://www.cygwin.com/> nebo [MinGW](http://www.mingw.org/) <http://www.mingw.org/>

- Základní použití (přepínače a argumenty) je u obou překladačů stejné

clang je kompatibilní s gcc

- Příklad použití

- compile: `gcc -c program.c -o program.o`
- link: `gcc program.o -o program`

Příklad součtu hodnot dvou proměnných

```

1 #include <stdio.h>
2
3 int main(void)
4 {
5     int var1;
6     int var2 = 10; /* inicializace hodnoty proměnné */
7     int sum;
8
9     var1 = 13;
10
11    sum = var1 + var2;
12
13    printf("The sum of %i and %i is %i\n", var1, var2, sum);
14
15    return 0;
16 }

```

- Proměnné `var1`, `var2` a `sum` reprezentují tři různá místa v paměti (automaticky přidělené), ve kterých jsou uloženy tři celočíselné hodnoty

Základní číselné typy

■ Celočíselné typy – `int`, `long`, `short`, `char`

`char` – celé číslo v rozsahu jednoho bajtu nebo také znak

- Velikost paměti alokované příslušnou (celo)číselnou proměnnou se může lišit dle architektury počítače nebo překladače

Typ `int` má zpravidla velikost 4 bajty a to i na 64-bitových systémech

- Aktuální velikost paměťové reprezentace lze zjistit operátorem `sizeof()`, kde argumentem je jméno typu nebo proměnné.

```
int i;
printf("%lu\n", sizeof(int));
printf("ui size: %lu\n", sizeof(i));
```

`lec01/types.c`

■ Neceločíselné typy – `float`, `double`

Jsou dané implementací, většinou dle standardu IEEE-754-1985

- `float` – 32-bit IEEE 754
- `double` – 64-bit IEEE 754

http://www.tutorialspoint.com/cprogramming/c_data_types.htm

Znak – `char`

- Znak je typ `char`
- Znak reprezentuje celé číslo (byte)

Kódování znaků (grafických symbolů), např. ASCII – American Standard Code for Information Interchange.

- Hodnotu znaku lze zapsat jako tzv. znakovou konstantu, např. `'a'`.

```
1 char c = 'a';
2
3 printf("The value is %i or as char '%c'\n", c, c);
```

`lec01/char.c`

```
clang char.c && ./a.out
The value is 97 or as char 'a'
```

- Pro řízení výstupních zařízení jsou definovány řídicí znaky

Tzv. [escape sequences](#)

- `\t` – tabulátor (tabular), `\n` – nový řádek (newline),
- `\a` – pípnutí (beep), `\b` – backspace, `\r` – carriage return,
- `\f` – form feed, `\v` – vertical space

Znaménkové a neznaménkové celočíselné typy

- Celočíselné typy kromě počtu bajtů rozlišujeme na

- `signed` – **znaménkový** (základní)
- `unsigned` – **neznaménkový**

Proměnná neznaménkového typu nemůže zobrazit záporné číslo

- Příklad (1 byte):

```
unsigned char: 0 až 255
signed char: -128 až 127
```

```
1 unsigned char uc = 127;
2 char su = 127;
3
4 printf("The value of uc=%i and su=%i\n", uc, su);
5 uc = uc + 2;
6 su = su + 2;
7 printf("The value of uc=%i and su=%i\n", uc, su);
```

`lec01/signed_unsigned_char.c`

Logický datový typ (Boolean) – `_Bool`

- Ve verzi **C99** je zaveden logický datový typ `_Bool`

```
_Bool logic_variable;
```

- Jako hodnota `true` je libovolná hodnota typu `int` různá od 0
- Dále můžeme využít hlavičkového souboru `stdbool.h`, kde je definován typ `bool` a hodnoty `true` a `false`

```
#define false 0
#define true 1

#define bool _Bool
```

- V původním (ANSI) C explicitní datový typ pro logickou hodnotu není definován.

- Můžeme však použít podobnou definici jako v `stdbool.h`

```
#define FALSE 0
#define TRUE 1
```

Rozsahy celočíselných typů

- Rozsahy celočíselných typů v C nejsou dány normou, ale implementací *Mohou se lišit implementací a prostředím 16 bitů vs 64 bitů*

- Norma garantuje, že pro rozsahy typů platí
 - `short ≤ int ≤ long`
 - `unsigned short ≤ unsigned ≤ unsigned long`

- Pokud chceme zajistit definovanou velikost můžeme použít definované typy například v hlavičkovém souboru `stdint.h`

IEEE Std 1003.1-2001

```
int8_t      uint8_t
int16_t     uint16_t
int32_t     uint32_t

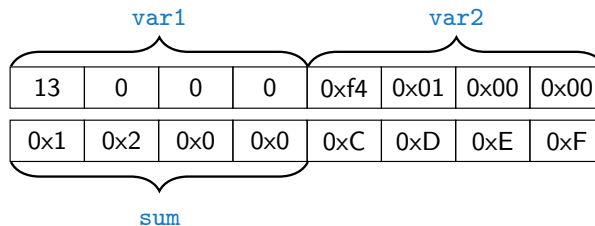
lec01/inttypes.c
```

<http://pubs.opengroup.org/onlinepubs/009695399/basedefs/stdint.h.html>

Přřazení, proměnné a paměť – Vizualizace int

```
1 int var1;
2 int var2;
3 int sum;
4
5 var1 = 13;
6 var2 = 500;
7
8 sum = var1 + var2;
```

- Proměnné typu `int` alokují 4 bajty *Zjistit velikost můžeme operátorem `sizeof(int)`*
- Obsah paměti není po alokaci definován



500 (dec) je 0x01F4 (hex)

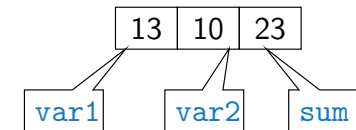
513 (dec) je 0x0201 (hex)

*V případě architektury Intel x86 a x86-64 jsou hodnoty uloženy v pořadí **little-endian***

Přřazení, proměnné a paměť – Vizualizace unsigned char

```
1 unsigned char var1;
2 unsigned char var2;
3 unsigned char sum;
4
5 var1 = 13;
6 var2 = 10;
7
8 sum = var1 + var2;
```

- Každá z proměnných alokuje právě 1 byte
- Obsah paměti není po alokaci definován
- Jméno proměnné „odkazuje“ na paměťové místo
- Hodnota proměnné je obsah paměťového místa



Výrazy

- Výraz** předepisuje výpočet hodnoty určitého vstupu
- Struktura výrazu obsahuje **operandy**, **operátory** a **závorky**
- Výraz může obsahovat
 - literály
 - proměnné
 - konstanty
 - unární a binární operátory
 - volání funkcí
 - závorky

- Pořadí operací předepsaných výrazem je dáno **prioritou** a **asociativitou** operátorů.

Příklad

```
10 + x * y // pořadí vyhodnocení 10 + (x * y)
10 + x + y // pořadí vyhodnocení (10 + x) + y
```

** má vyšší prioritu než +
+ je asociativní zleva*

Základní rozdělení operátorů

- Operátory jsou vyhrazené znaky (nebo posloupnost znaků) pro zápis výrazů
- Můžeme rozlišit čtyři základní typy binárních operátorů
 - Aritmetické** operátory – sčítání, odčítání, násobení, dělení
 - Relační** operátory – porovnání hodnot (menší, větší, ...)
 - Logické** operátory – logický součet a součin
 - Operátor přiřazení** - na levé straně operátoru `=` je proměnná
- Unární operátory
 - indikující kladnou/zápornou hodnotu: `+` a `-`
operátor – modifikuje znaménko výrazu za ním
 - modifikující proměnou: `++` a `--`
 - logický operátor doplněk: `!`
- Ternární operátor – podmíněné přiřazení hodnoty

Základní aritmetické výrazy

- Pro operandy číselných typů `int` a `double` jsou definovány operátory *Ale také pro `char`, `short`, `float`*
 - unární operátor změna znaménka `-`
 - binární sčítání `+` a odčítání `-`
 - binární násobení `*` a dělení `/`
- Pro operandy celočíselných typů pak dále
 - binární zbytek po dělení `%`
- Pro oba operandy stejného typu je výsledek aritmetické operace stejného typu
- V případě kombinace typů `int` a `double`, se `int` převede na `double` a výsledek je hodnota typu `double`. *Implicitní typová konverze*
- Dělení operandů typu `int` je celá část podílu
Např. $7/3$ je 2 a $-7/3$ je -2
- Pro zbytek po dělení platí $x\%y = x - (x/y) * y$
Např. $7 \% 3$ je 1 $-7 \% 3$ je -1 $7 \% -3$ je 1 $-7 \% -3$ je -1
*Pro záporné operandy je v C99 výsledek celočíselného dělení blíže 0, platí $(a/b)*b + a\%b = a$. Pro starší verze C závisí výsledek na překladači.*
Další aritmetické operátory příště.

Proměnné, operátor přiřazení a příkaz přiřazení

- Proměnné definujeme uvedením typu a jména proměnné
 - Jména proměnných volíme malá písmena
 - Víceslovná jména zapisujeme s podtržítkem `_`
Nebo volíme `CamelCase`
 - Proměnné definujeme na samostatném řádku

```
int n;  
int number_of_items;
```
- Přiřazení je nastavení hodnoty proměnné, tj. uložení definované hodnoty na místo v paměti, kterou proměnná reprezentuje
- Tvar **přiřazovacího operátoru**
 $\langle \text{proměnná} \rangle = \langle \text{výraz} \rangle$
Výraz je literál, proměnná, volání funkce, ...
- Příkaz přiřazení** se skládá z operátoru přiřazení `=` a ;
 - Levá strana přiřazení musí být **l-value – location-value, left-value**
Tj. musí reprezentovat paměťové místo pro uložení výsledku.
 - Přiřazení je výraz a můžeme jej použít všude, kde je dovolen výraz příslušného typu

Příklad – Aritmetické operátory 1/2

```
1 int a = 10;  
2 int b = 3;  
3 int c = 4;  
4 int d = 5;  
5 int result;  
6  
7 result = a - b; // rozdíl  
8 printf("a - b = %i\n", result);  
9  
10 result = a * b; // násobení  
11 printf("a * b = %i\n", result);  
12  
13 result = a / b; // celocíselne deleni  
14 printf("a / b = %i\n", result);  
15  
16 result = a + b * c; // prioritá operatoru  
17 printf("a + b * c = %i\n", result);  
18  
19 printf("a * b + c * d = %i\n", a * b + c * d); // -> 50  
20 printf("(a * b) + (c * d) = %i\n", (a * b) + (c * d)); // -> 50  
21 printf("a * (b + c) * d = %i\n", a * (b + c) * d); // -> 350
```

lec01/arithmetick_operators.c

Příklad – Aritmetické operátory 2/2

```

1  #include <stdio.h>
2
3  int main(void)
4  {
5      int x1 = 1;
6      double y1 = 2.2357;
7      float x2 = 2.5343f;
8      double y2 = 2;
9
10     printf("P1 = (%i, %f)\n", x1, y1);
11     printf("P1 = (%i, %i)\n", x1, (int)y1);
12     printf("P1 = (%f, %f)\n", (double)x1, (double)y1);
13     printf("P1 = (%.3f, %.3f)\n", (double)x1, (double)y1);
14
15     printf("P2 = (%f, %f)\n", x2, y2);
16
17     double dx = (x1 - x2); // implicitni konverze na float, resp.
18     double dy = (y1 - y2);
19
20     printf("(P1 - P2)=(%.3f, %0.3f)\n", dx, dy);
21     printf("|P1 - P2|^2=%.2f\n", dx * dx + dy * dy);
22     return 0;
23 }

```

lec01/points.c

Formátovaný výstup – printf()

- Číselné hodnoty lze tisknout (vypsat) na standardní výstup prostřednictvím funkce `printf()`

man printf, resp. man 3 printf

- Argumentem funkce je textový řídicí řetězec formátování výstupu
- Řídicí řetězec formátu je uvozen znakem `'%'`
- Znakové posloupností (nezačínající `%`) se vypíše tak jak jsou uvedeny
- Základní řídicí řetězce pro výpis hodnot jednotlivých typů

char	%c
_Bool	%i, %u
int	%i, %x, %o
float	%f, %e, %g, %a
double	%f, %e, %g, %a

- Dále je možné specifikovat počet vypsaných míst, zarovnání vlevo (vpravo), atd.

Více na cvičení a v domácích úkolech.

Standardní výstup a vstup

- Spuštěný program v prostředí operačního systému má přiřazený znakově orientovaný standardní vstup (`stdin`) a výstup (`stdout`)
Výjimkou jsou zpravidla programy pro MCU bez OS.
- Program může prostřednictvím `stdout` a `stdin` komunikovat s uživatelem
- Základní funkce pro znakový výstup je `putchar()` a pro vstup `getchar()` definované ve standardní knihovně `stdio.h`.
- Pro načítání číselných hodnot lze využít funkci `scanf()`
- Formátovaný výstup je možné tisknout funkce `printf()`, např. číselné hodnoty

Jedná se o knihovní funkce, ze standardní knihovny. Jména funkcí nejsou klíčová slova jazyka C.

Formátovaný vstup – scanf()

- Číselné hodnoty ze standardního vstupu lze načíst funkcí `scanf()`
man scanf, resp. man 3 scanf
- Argumentem je textový řídicí řetězec *Syntax podobný příkazu printf()*
- Je nutné předat paměťové místo pro uložení hodnoty
- Příklad načtení hodnoty celého čísla a hodnoty typu `double`

```

1  #include <stdio.h>
2
3  int main(void)
4  {
5      int i;
6      double d;
7
8      printf("Enter int value: ");
9      scanf("%i", &i); /* operator & vraci adresu
10     promenne i */
11
12     printf("Enter a double value: ");
13     scanf("%lf", &d);
14     printf("You entered %02i and %0.1f\n", i, d);
15
16     return 0;
17 }

```

lec01/scanf.c

Část III

Část 3 – Zadání 1. domácího úkolu (HW01)

Shrnutí přednášky

Zadání 1. domácího úkolu HW01

Téma: Načítání vstupu, výpočet a výstup

Povinné zadání: **2b**; Volitelné zadání: **není**; Bonusové zadání: **není**

- **Motivace:** Získat představu o interakci uživatele s programem
- **Cíl:** Osvojit si načítání vstupu, formátovaného výstupu a základní posloupnosti příkazů
- **Zadání:** <https://cw.fel.cvut.cz/wiki/courses/b0b36prp/hw/hw01>
 - Načítání celých čísel ze standardního vstup
 - Výpis čísel v dekadické a šestnáctkové soustavě
 - Provedení základní aritmetických operací s načtenými čísly
 - Výpočet podílu a průměrné hodnoty čísel
 - Dodržení správného formátování výstupu

Použijte **hex** zobrazení výstupu – **hexdump -C**
- **Termín odevzdání:** **15.10.2016, 23:59:59 PDT**

PDT – Pacific Daylight Time

Diskutovaná témata

- Informace o předmětu
- Základy programování v C
 - Program, zdrojové soubory a kompilace programu
 - Struktura zdrojového souboru a zápis programu
 - Proměnné, základní číselné typy
 - Proměnné, přiřazení a paměť
 - Základní výrazy
 - Standardní vstup a výstup programu
 - Formátovaný vstup a výstup
- **Příště: Zápis programu v C a základní řídicí struktury**