

Druhé počítačové cvičení

Základy programování (IZP)

Brno University of Technology, Faculty of Information Technology
Božetěchova 1/2, 612 66 Brno - Královo Pole
Petr Veigend, veigend@fit.vut.cz



- **Programování v Linuxu, online**
- **Úhlednost zdrojového kódu**
- **Opakování základních pojmů**
- **Jednoduché programy**
- **Cykly**
- **Pole**

PROGRAMOVÁNÍ V LINUXU

- V Linuxu otevřete program Terminal
 - V učebně: Applications → System Tools → Terminal
 - Windows: program **putty**
- Po spuštění terminálu uvidíte

`login@název počítače $`
- Lze použít Visual Studio Code a další editory a prostředí
 - [Jak je nainstalovat najdete online](#)
- Ve Windows lze použít **WSL** pro emulaci Linuxu (programování potom funguje stejně jako v Linuxu)

- Ve Windows spusťte program PuTTY
- Nastavení PuTTY

Category: Terminal → Features → zaškrtnout
Disable application keypad mode (NumLock)

Category: Session

- **Host name:** merlin.fit.vutbr.cz, **protokol:** **SSH**
- **Přihlašovací jméno:** **váš login FIT**
- **Saved Sessions:** **Merlin**, potom **Save**
- **Heslo:** **heslo, kterým se přihlašujete do WIS**

- Windows
 - WinSCP
- Linux
 - Příkaz scp
 - GUI (ve většině distribucí)
- WIS
 - FTP klient (sekce Ostatní, server eva)
 - Doporučeno pro začátečníky

- **ls** – výpis obsahu adresáře
- **mkdir** – vytvoření nového adresáře

```
mkdir název_adresáře
```

- **cd** – změna adresáře

```
cd název_adresáře
```

- **rm, rmdir** – odstranění souboru, adresáře

```
rm název_souboru
```

```
rmdir název_adresáře
```

- **mc** – Midnight Commander
(grafický správce souborů pro systémy unixového typu)
- Další informace:

```
man název_příkazu
```

- Vytvoříme adresář `izp` (záleží na velikosti písmen)

```
mkdir izp
```

- Přejdeme do něj

```
cd izp
```

- Vytvoříme adresář `hello` a přejdeme do něj

```
mkdir hello
```

```
cd hello
```

- a vytvoříme v něm soubor `hello.c` (pomocí editoru `nano`)

```
nano hello.c
```

- Pomocí editoru nano

```
#include <stdio.h>
int main()
{
    printf("Hello world!\n");
    return 0;
}
```

Uložení změn: CTRL + O

Ukončení editoru: CTRL + X

- Kontrola: `cat hello.c`
 - Zobrazí obsah souboru hello.c

- Nejdříve si zkontrolujeme, zda máme ve složce `izp/hello` soubor `hello.c`

```
ls
```

- **Překlad** vytvoří spustitelný soubor `hello`

```
gcc -o hello hello.c
```

- **Spuštění** programu `hello`

```
$ ./hello
```

- Tečka – znamená aktuální adresář (tedy `~/hello/izp`)
 - Tilda (vlnka, `~`) – domovský adresář
- Lomítko – odděluje název adresáře a souboru

- Na minulém cvičení jsme překládali program

```
gcc -o hello hello.c
```

- Fungovalo to, ALE ...
 - Překladač nehlídá spoustu chyb a nevypisuje spoustu varování (např. neinicializované proměnné, přiřazení místo porovnání, atp.)
- Od teď budeme překládat takto

```
gcc -Wall -Wextra -pedantic hello.c -o  
hello
```

- Další parametry přidáme později.

- Ing. Dolejška připravil online prostředí pro programování v rámci IZP
 - <https://hub.bazar.nesad.fit.vutbr.cz/hub/home>
- Pokud budete chtít prostředí použít, můžete, používejte Empty environment.
- Překlad probíhá stejně jako na předchozích slajdech.
- POZOR:
 - Data v rámci tohoto online prostředí se **neukládají**
 - Když server vypnete, nebo uplyne 12 hodin, o data přijdete → **na práci na projektech je potřeba mít lokální prostředí**

- Základní pojmy
 - **Proměnná**
 - **Konstanta**
 - **Přiřazení (=)**
 - **Deklarace**
 - **Inicializace**
 - **Příkaz**

- Základní datové typy, které budeme používat
 - **int** – celá čísla (minule)
 - **char** – znaky
- Myslíte, že spolu tyto typy nějak souvisí ?

- Základní datové typy, které budeme používat
 - **int** – celá čísla (minule)
 - **char** – znaky
- Myslíte, že spolu tyto typy nějak souvisí ?
- Další datové typy, které můžeme použít
 - Desetinná čísla: `float`, **double**
 - Pole (viz dále)
 - Vlastní (později)

- K načítání vstupů se používá funkce `scanf`
`scanf ("formátovací řetězec", &proměnná)`
- Formátovací řetězec indikuje formát dat, která načítáme např. `%f`, `%d`
- Proměnná, do které budou zadaná data uložena
 - Znak `&`: chceme adresu, na kterou ukládáme
 - Pozor na **řetězce/pole** (viz dále)
- Příklad:

```
int celeCislo; // kam ukládáme
scanf ("%d", &celeCislo); // načtení
printf ("%d\n", celeCislo); // výpis
```

- Minule jsme počítali diskriminant (D), dnes ho odmocníme a vypočítáme kořeny kvadratické rovnice
 - Odmocnina: funkce `sqrt`, `#include<math.h>`, `-lm`
 - $$x_{1,2} = \frac{-b \pm \sqrt{D}}{2a} = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

v C: `(-b+sqrt(b*b-4*a*c))/(2*a)`

```
if (podmínka) { // == rovnost, != nerovnost, >, <
// tělo podmínky
}
```

- Bude \sqrt{D} vždy celé číslo? **double**
- Formátovací řetězec pro **double**: **%lf**

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

int main() {
    // vypocet
}
```

- Překlad zdrojového souboru (např. koreny.c)

```
gcc -Wall -Wextra -pedantic koreny.c -o
koreny -lm
```

- Spuštění `./koreny`

- Vytvořte proměnnou typu **char** (znak) a načtěte do ní znak
 - funkce **getchar()**
- Podmínka pro znaky např.

```
char a= 'x' // jednoduché úvozovky!!!  
char b= getchar();  
if (a == 'y') {  
    // tělo podmínky  
    // pozor na jednoduché úvozovky  
}
```

- ASCII tabulka

dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char	dec	hex	oct	char
0	0	000	NULL	32	20	040	space	64	40	100	@	96	60	140	`
1	1	001	SOH	33	21	041	!	65	41	101	A	97	61	141	a
2	2	002	STX	34	22	042	"	66	42	102	B	98	62	142	b
3	3	003	ETX	35	23	043	#	67	43	103	C	99	63	143	c
4	4	004	EOT	36	24	044	\$	68	44	104	D	100	64	144	d
5	5	005	ENQ	37	25	045	%	69	45	105	E	101	65	145	e
6	6	006	ACK	38	26	046	&	70	46	106	F	102	66	146	f
7	7	007	BEL	39	27	047	'	71	47	107	G	103	67	147	g
8	8	010	BS	40	28	050	(72	48	110	H	104	68	150	h
9	9	011	TAB	41	29	051)	73	49	111	I	105	69	151	i
10	a	012	LF	42	2a	052	*	74	4a	112	J	106	6a	152	j
11	b	013	VT	43	2b	053	+	75	4b	113	K	107	6b	153	k
12	c	014	FF	44	2c	054	,	76	4c	114	L	108	6c	154	l
13	d	015	CR	45	2d	055	-	77	4d	115	M	109	6d	155	m
14	e	016	SO	46	2e	056	.	78	4e	116	N	110	6e	156	n
15	f	017	SI	47	2f	057	/	79	4f	117	O	111	6f	157	o
16	10	020	DLE	48	30	060	0	80	50	120	P	112	70	160	p
17	11	021	DC1	49	31	061	1	81	51	121	Q	113	71	161	q
18	12	022	DC2	50	32	062	2	82	52	122	R	114	72	162	r
19	13	023	DC3	51	33	063	3	83	53	123	S	115	73	163	s
20	14	024	DC4	52	34	064	4	84	54	124	T	116	74	164	t
21	15	025	NAK	53	35	065	5	85	55	125	U	117	75	165	u
22	16	026	SYN	54	36	066	6	86	56	126	V	118	76	166	v
23	17	027	ETB	55	37	067	7	87	57	127	W	119	77	167	w
24	18	030	CAN	56	38	070	8	88	58	130	X	120	78	170	x
25	19	031	EM	57	39	071	9	89	59	131	Y	121	79	171	y
26	1a	032	SUB	58	3a	072	:	90	5a	132	Z	122	7a	172	z
27	1b	033	ESC	59	3b	073	;	91	5b	133	[123	7b	173	{
28	1c	034	FS	60	3c	074	<	92	5c	134	\	124	7c	174	
29	1d	035	GS	61	3d	075	=	93	5d	135]	125	7d	175	}
30	1e	036	RS	62	3e	076	>	94	5e	136	^	126	7e	176	~
31	1f	037	US	63	3f	077	?	95	5f	137	_	127	7f	177	DEL

- Vytvořte proměnnou typu **char** (znak) a načtěte do ní znak
 - funkce **getchar()**
- Podmínka pro znaky např.

```
char a = 'x' // jednoduché úvozovky!!!  
char b = getchar();  
if (a == 'y') {  
    // tělo podmínky  
    // pozor na jednoduché úvozovky}
```

- ASCII tabulka

- Načtěte **tři** čísla funkcí **scanf** a pro každé určete, zda je sudé

```
for (int i = 0; i < ??; i++)  
{  
    // proměnná pro uložení celého čísla  
    // scanf  
    // podmínka pro porovnání, operace %  
}
```

- **Cyklus** (počítaný, nepočítaný)
 - Cyklem myslíme opakování určité části programu
 - **Počítaný**: **známý** počet průchodů (**for**)
 - **Nepočítaný**: **neznámý** počet průchodů (**while**, **do-while**)

```
for (int i = 0; i < 10; i++)  
{  
    printf("%d\n", i);  
}
```

- Načtete **počet čísel**, která budete testovat na sudost
- Funkcí **scanf** tato čísla poté načtete a pro každé určete, zda je sudé

```
for (int i = 0; i < ??; i++)  
{  
    // proměnná pro uložení celého čísla  
    // scanf  
    // podmínka pro porovnání, operace %  
}
```

- Napište program, který bude počítat **faktoriál** zadaného čísla
- Příklad: $5! = 5 \times 4 \times 3 \times 2 \times 1$

Skutečná adresa	1634	1635	1636	1637	1638	1639
Index	0	1	2	3	4	5
Hodnota	10	20	30	40	50	60

Skutečná adresa	1634	1635	1636	1637	1638	1639
Index	0	1	2	3	4	5
Hodnota	10	20	30	40	50	60

- **Pole:** prvky stejného typu, spojitě místo v paměti

Skutečná adresa	1634	1635	1636	1637	1638	1639
Index	0	1	2	3	4	5
Hodnota	10	20	30	40	50	60

- **Pole:** prvky stejného typu, spojitě místo v paměti
- **Deklarace staticky:** `int moje_pole[6];`

Skutečná adresa	1634	1635	1636	1637	1638	1639
Index	0	1	2	3	4	5
Hodnota	10	20	30	40	50	60

- **Pole:** prvky stejného typu, spojitě místo v paměti
- **Deklarace staticky:** `int moje_pole[6];`
- **Přístup k položkám ([]):** `moje_pole[1] = 5;`

Skutečná adresa	1634	1635	1636	1637	1638	1639
Index	0	1	2	3	4	5
Hodnota	10	5	30	40	50	60

- **Pole:** prvky stejného typu, spojitě místo v paměti
- **Deklarace staticky:** `int moje_pole[6];`
- **Přístup k položkám ([]):** `moje_pole[1] = 5;`

Skutečná adresa	1634	1635	1636	1637	1638	1639
Index	0	1	2	3	4	5
Hodnota	10	5	30	40	50	60

- **Pole:** prvky stejného typu, spojitě místo v paměti
- **Deklarace staticky:** `int moje_pole[6];`
- **Přístup k položkám ([]):** `moje_pole[1] = 5;`
- **Např. pole typu řetězec**

```
char *msg[3] = {"warning", "error", "fatal"};
int problem_level = 1; // co by se vypsallo pro 3
printf("%s: oops!\n", msg[problem_level]);
```

- Načtete 5 čísel do pole a vypíšete je v opačném pořadí.
- Pole: `int pole[5];`
- Načtení čísel: funkce `scanf`
- Jak vypíšeme obsah pole v opačném pořadí?
 - Inicializace řídicí proměnné cyklu
 - Jak se bude řídicí proměnná měnit v každé iteraci?

- Načtěte 5 čísel do pole a najděte:
 - Jejich maximum
 - Jejich minimum
 - Jejich průměr

- Načtěte dvě čísla typu **float**, která budou reprezentovat souřadnice bodů
 - Nezapomeňte vytvořit proměnné pro uložení čísel
 - Funkce **scanf**, dva formátovací řetězce %f
 - Výpis pomocí **printf (%f)** ve tvaru
`[prvníCislo, druhéCislo]`

- Vytvořte program, který spočítá spotřebu pohonných hmot
- Uživatel bude muset zadat čtyři parametry
 - Počáteční stav nádrže [l]
 - Počáteční stav tachometru [km]
 - Koncový stav nádrže [l]
 - Koncový stav tachometru [km]
- Výstupem bude průměrná spotřeba paliva na 100 km

- Napište program, který převede zadaný počet hodin a minut na sekundy.
 - Ve výstupu musí být patrný počet hodin a sekund a převedená hodnota.

- Napište program, který zadaný počet sekund převede na hodiny, minuty a sekundy.
 - Ve výstupu musí být patrný počet sekund a převedená hodnota.

Děkuji za pozornost