

Organizace předmětu, vývojové nástroje, základy programování a podmínky

IZP-cv01

Ing. Jakub Husa

Vysoké Učení Technické v Brně, Fakulta informačních technologií
Božetěchova 1/2. 612 66 Brno - Královo Pole
ihusa@fit.vut.cz



23. října 2023

Organizace předmětu

Garantem předmětu je docentka **Jitka Kreslíková** (kreslika@fit.vut.cz):

- Zodpovídá za vyučovanou látku, zkoušky, a organizaci předmětu.

Zástupcem garanta je doktor **Aleš Smrčka** (smrcka@fit.vut.cz):

- Zodpovídá za náplň cvičení a projektů.

Výuku cvičení zajišťuje cca 15 různých cvičících:

- Vaším cvičícím jsem já, inženýr **Jakub Husa** (ihusa@fit.vut.cz).
- Konzultace nabízím po dohodě e-mailem, v **kanceláři L307**.

Vaše přítomnost ve výuce je **ZCELA DOBROVOLNÁ**:

- Svoji absenci nikomu neomlouváte – můžete ale přijít o body.
- Cvičení si **stejný týden** můžete nahradit v jiném termínu, nejlépe u stejného cvičícího (Po 16:00 a 18:00, Ut 16:00 a 18:00, St 18:00, Čt 16:00, vždy v **učebně N105**).
- Předchozí týdny si můžete nahradit pouze při překážce ve studiu.

<https://www.vut.cz/studis> -> Řízení a žádosti -> Založit žádost
-> Druh: Oznámení -> Žádost: Zaevidování překážky ve studiu



Každý týden budete mít:

- Dvě **přednášky** (2h a 1h).
- Jedno **demonstrační** cvičení (1h).
- Jedno **laboratorní** cvičení (2h).

Během semestru budete mít:

- Dva samostatné domácí **projekty**.
- Jednu **půlsestrální** zkoušku.
- Jednu **semestrální** zkoušku.

Všechny potřebné informace najdete v informačním systému:

- Informace o předmětu a vaše bodové hodnocení uvidíte ve **Studisu**.
- Studijní materiály a organizační informace najdete v **E-learningu (Moodle)**.
- Pozor – ostatní předměty mohou mít informace umístěny úplně jinde!

Pro komunikaci vždy používejte svůj **školní e-mail**:

- Pozor – **discord server** od **studentské unie** **NENÍ** oficiální komunikační nástroj!

Slidy ze cvičení, na které se právě díváte, najdete na mojí stránce:

- www.fit.vut.cz -> hledat "Jakub Husa"-> Výuka -> Slidy ze cvičení IZP

Za semestr můžete z IZP získat až 100 bodů:

- Přednášky a demonstrační cvičení nejsou bodované.
- Laboratorní cvičení jsou každé za 1 bod (celkem 10).
- Půlsestrální zkouška je za 12 bodů, trvá 40 minut, máte na ni jeden pokus a bude se psát 10. listopadu od 18:00 (pátek, 8. týden).
- Semestrální zkouška je za 54 bodů, trvá 90 minut, máte na ni tři pokusy a bude se psát začátkem příštího roku (termíny zatím nejsou rozhodnuté).

První projekt je na práci s textem a je za něj 10 bodů:

- Hodnocena je pouze funkčnost – automaticky, doktorem Smrčkou.

Druhý projekt je na práci s datovými strukturami a je za něj 14 bodů:

- Hodnoceny jsou funkčnost, obhajoba, a kvalita zdrojového kódu (komentáře, přehlednost, dekompozice, identifikátory, ...)
- **Obhajoba** – cca 5 minut vysvětlujete a odpovídáte na otázky jak vámi odevzdaný zdrojový kód funguje, abyste prokázali že jste jeho autorem.
- Obhajoby jsou 12. týden místo laboratorního cvičení.

Zadání projektů uvidíte **E-learningu** a budou pro všechny studenty stejná:

- Pozor – zatím vidíte pouze zadání z loňského roku!
- Pro dotazy k projektům jsou v E-learningu zavedena fóra.
- Pozor – odpovědi zveřejněné na fórech **jsou závazné pro všechny studenty!**

Projekty budete psát v **jazyku C**, který se spolu tento semestr budeme učit:

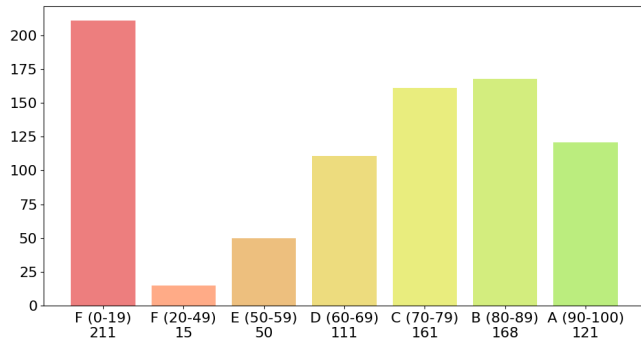
- Formálně – vaše projekty musejí splňovat normu **ISO/IEC 9899**.
- Prakticky – vaše projekty musejí fungovat na studentském serveru **merlin.fit.vutbr.cz**, který bude sloužit jako váš referenční stroj.

Všechny odevzdávané soubory jsou každý-s-každým **automaticky kontrolovány** na podobnost se soubory od všech ostatních studentů:

- Přílišná podobnost souborů u vás způsobí podezření z **plagiátorství**.
- Pokud se stanete podezřelými, pozve si vás doktor Smrčka.
- Pokud mu situaci nedokážete vysvětlit, pozve si vás **disciplinární komise**.
- Pokud jí situaci nedokážete vysvětlit, komise **UKONČÍ VAŠE STUDIUM**.

Pro úspěšné zvládnutí předmětu potřebujete získat alespoň:

- 6 bodů ze cvičení (max 10).
- 1 bod z 1. projektu (max 10).
- 0 bodů z půlsemestrálky (max 12).
- 1 bod z 2. projektu (max 14).
- 23 bodů před zkouškou (zápočet).
- 23 bodů ze zkoušky (max 54).
- 50 bodů celkem.



Minulý rok předmět nezvládlo 27% studentů,
93% z nich proto že nezískali zápočet!

Vývojové nástroje

Jazyk C je **imperativní** a **kompilovaný** programovací jazyk:

- **Imperativní** – program je tvořen posloupností **příkazů** definujících postup řešení nějakého problému (**algoritmus**).
- **Kompilovaný** – zdrojový kód programu nemůžeme spustit na přímo, ale musíme ho vždy nejprve přeložit na **spustitelný soubor**.

K programování budeme potřebovat nějaký **editor** a **překladač**:

- **Editor** – slouží k editování souboru do kterého budeme psát **zdrojový kód**.
- **Překladač** – slouží k přeložení zdrojového kódu na **spustitelný soubor**.

Programy se budeme učit psát tak aby byly **multiplatformní**:

- **Multiplatformní** – program který jde bez úprav přeložit a spustit na různých počítačích s různými **operačními systémy**.
- V předmětu IZP budeme uvažovat pouze operační systémy **Windows** a **Unix**.

Ve Windows máme k dispozici **vývojová prostředí** poskytující editor i překladač:

- Osobně budu ukazovat prostředí **Code::Blocks**, ale **volba je na vás**.
- Na školních počítačích ho najdete **Q: -> netapp -> CodeBlocks**.
- Na vlastním počítači si ho můžete **stáhnout** z **www.codeblocks.org**
- Programovat můžete i on-line, například: **tady**, **zde**, nebo **tamhle**.

Ve vývojovém prostředí program zakládáme jako **projekt**, který kromě **zdrojového kódu** obsahuje také parametry překladač a další nastavení:

- V předmětu IZP budeme tvořit pouze **konzolové aplikace**.
File -> New -> Project... -> Console application -> C
- Projektu dáme jméno, vybereme si kam ho chceme uložit (na školních počítačích vždy na disk **P:**) a potvrdíme **Next -> Finish**.
- Projekt se objeví v levé straně okna **Workspace -> JMENO -> Sources -> main.c**.
- Program přeložíme a spustíme **Build -> Build and run**, nebo klávesou **F9**.
- Spuštění programu otevře **konzoli** (černé okno), které zavřeme klávesou **Enter**.
- Pokud se překlad nepodařil, překladač v dolním okně vypíše v čem byla chyba.

Zdrojový kód programu se skládá z podprogramů (funkcí):

- Vykonávání programu vždy začíná **hlavní funkcí** jménem **main**.
- Další funkce můžeme importovat ze **standardních knihoven** jazyka C.
- Seznam standardních knihoven a jejich funkcí **najdeme na referenci**.

Ukázkový program – **Hello, World!**:

```
1 #include <stdio.h>           //vlozeni knihovny pro vstup a vystup
2 #include <stdlib.h>         //vlozeni knihovny zakladnich funkci
3                             //prazdny radek (jen pro prehlednost)
4 int main()                 //hlavicka funkce main (zacatek programu)
5 {                           //zacatek tela funkce main
6     printf("Hello, World!\n"); //knihovni funkce pro vypis textu
7     return 0;              //prikaz ukonceni funkce (konec programu)
8 }                           //konec tela funkce main
```

Dvojité lomítko (//) v jazyku C označuje jednořádkový **komentář**:

- Komentáře vysvětlují chování kódu, ale na jeho funkcionalitu nemají vliv.
- Víceřádkové komentáře se ohraničují značkami (/*) a (*).

Fakulta vám pro studijní účely poskytuje přístup na server merlin.fit.vutbr.cz:

- Server se používá jako **referenční stroj** pro hodnocení vašich projektů.

Operačním systémem serveru je **CentOS (Unix)**:

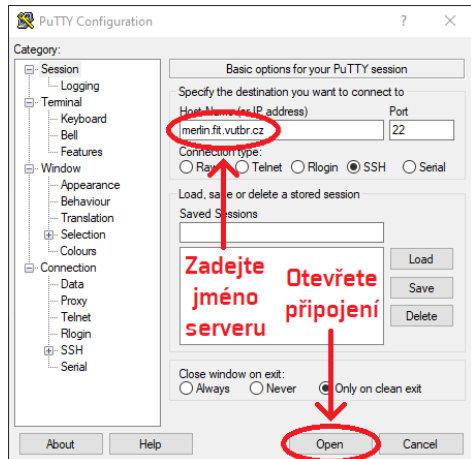
- Na stránkách fakulty najdete **návod pro práci s Unixem** pro začínající studenty.

Z Windows se na server připojíte přes **PuTTY**:

- Na školních počítačích program najdete ve složce **Q**: -> **netapp** -> **PUTTY.exe**
- Na vlastním počítači si program můžete **stáhnout** ze stránky www.putty.org

Z Unixu se na server připojíte **protokolem SSH**:

- `ssh xlogin00@merlin.fit.vutbr.cz`

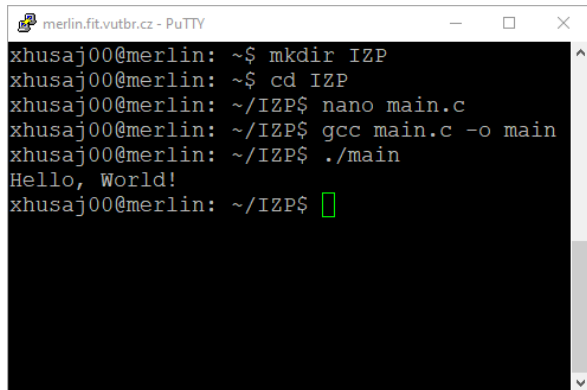


Operační systém Unix ovládáme psaním **příkazů** na **příkazovou řádku**:

- Jednotlivé příkazy spouštějí **systemové nástroje** nebo **nainstalované programy**.
- Některé příkazy mohou mít **parametry** specifikující jejich činnost.

Některé základní příkazy a programy:

- **mkdir NAZEV** – vytvoř složku.
- cd NAZEV** – otevři složku.
- cd ..** – zavři složku.
- rmdir NAZEV** – odstraň složku.
- ls** – vypiš obsah složky.
- **nano NAZEV** – vytvoř a otevři soubor.
- rm NAZEV** – odstraň soubor.
- gcc NAZEV** – přelož soubor.
- ./NAZEV** – spusť soubor.
- exit** – odhlas se ze serveru.



```
merlin.fit.vutbr.cz - PuTTY
xhusaj00@merlin: ~$ mkdir IZP
xhusaj00@merlin: ~$ cd IZP
xhusaj00@merlin: ~/IZP$ nano main.c
xhusaj00@merlin: ~/IZP$ gcc main.c -o main
xhusaj00@merlin: ~/IZP$ ./main
Hello, World!
xhusaj00@merlin: ~/IZP$
```

Na příkazové řádce nevytváříme **projekt** ale přímo **zdrojový soubor**:

- **Copy** v Unixu provedeme označením textu, **Paste** pravým tlačítkem myši.
- Zkratka **Ctrl+c** v Unixu ukončuje běžící program (**alt+F4** ve Windows).

Zdrojový soubor (**main.c**) můžeme otevřít textovým editorem **Nano**:

- `nano main.c`
- Soubor uložíme zkratkou **ctrl+o**, uložení potvrdíme klávesou **Enter**, a soubor zavřeme zkratkou **ctrl+x**.
- **Zvýrazňování syntaxe** jde nastavit.

Soubor přeložíme překladačem **GCC**:

- `gcc main.c -o main`
- Pokud překladač nevypisuje chybu tak se překlad podařil.

```

merlin.fit.vutbr.cz - PuTTY
GNU nano 2.3.1      Soubor: main.c      Změněno
#include <stdio.h>
#include <stdlib.h>

int main()
{
    printf("Hello world!\n");
    return 0;
}
[ Řádek 9/9 (100%), Sloupec 1/1 (100%), Znak 99/99 (1 )
^G Získat^O Uložit^R Otevří^Y Předch^K Vyjmou^C Poloha
^X Ukonči^J Zarovn^W Kde je^V Další ^U Zrušit^T Pravopi
    
```

Školní počítače soubory neukládají lokálně, ale na dva sdílené **souborové servery**:

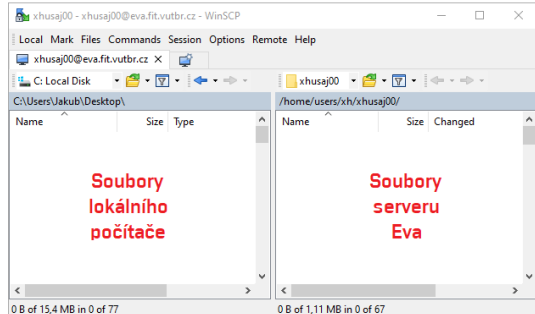
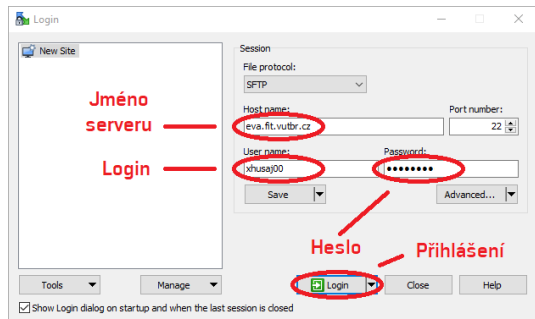
- Unixové soubory (ty ze serveru **Merlin**) najdete na serveru **eva.fit.vutbr.cz**.

Z Windows se k vašim souborům dostanete programem **WinSCP**.

- Na školních počítačích ho najdete ve složce **Q**: -> **netapp** -> **WinSCP**.
- Na vlastním počítači si ho můžete **stáhnout** z **www.winscp.net**.

V Unixu můžete použít **protokol SCP**:

- `scp ZDROJ LOGIN@SERVER:CESTA/CIL`
- `scp LOGIN@SERVER:CESTA/ZDROJ CIL`



Školní počítače soubory neukládají lokálně, ale na dva sdílené **souborové servery**:

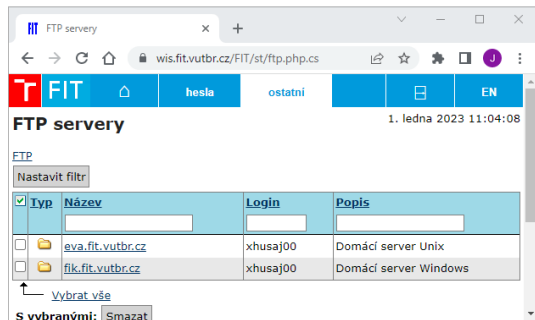
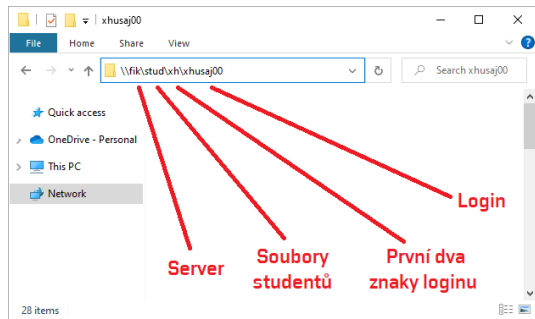
- Windowsové soubory (ty na disku **P:**) najdete na serveru **fik.fit.vutbr.cz**.

Pokud jste ve školní síti, k souborům se dostanete **průzkumníkem souborů** přes **Microsoft Active Directory**:

- Otevřete adresář **\\fik\stud\xxx\yyy** kde **xxx** jsou první dva znaky vašeho loginu, **yyy** je váš login, a přihlaste se.

Pokud ve školní síti nejste, k souborům se dostanete přes starý informační systém FIT:

- **wis.fit.vutbr.cz** -> **Ostatní** -> **FTP klient**



Základy programování

Data v programu ukládáme do **proměnných**:

- Každá proměnná musí mít určen nějaký **datový typ** a **identifikátor**.
- **Datový typ** – udává rozsah možných hodnot a dostupné operace.
- **Identifikátor** – označuje místo v paměti na kterém jsou data uložena.

Identifikátory musejí začínat písmenem nebo podtržítkem (**a-z, A-Z, _**) a mohou pokračovat libovolnou kombinací písmen, číslic a podtržítok (**a-z, A-Z, 0-9, _**):

- V jednom **bloku kódu** nelze mít dvě proměnné se stejným identifikátorem.
- Identifikátor se nesmí shodovat s žádným z **klíčových slov**.
- **Celá čísla** v rozsahu od -2^{31} do $2^{31}-1$ ukládáme do datového typu – **int** – (**integer**).

```
1 int x; //cele cislo jmenem "x"
2 int x1; //cele cislo jmenem "x1"
3 int x_1; //cele cislo jmenem "x_1"
4 int x 1; //CHYBA -- jmeno promenne nesmi obsahovat mezery
5 int 1x; //CHYBA -- jmeno promenne nesmi zacinat cislici
6 int int; //CHYBA -- jmeno promenne nesmi byt klicovym slovem
7 int x; //CHYBA -- promenna jmenem "x" jiz existuje
```

Hodnotu proměnné definujeme pomocí **operátoru přiřazení (=)**:

- Hodnotu proměnné můžeme definovat už při jejím vytvoření.

```
1 int a = 10;           //vytvor cele cislo "a" s hodnotou 10
2 a = 20;              //hodnotu promenne "a" zmen na 20
```

Hodnoty zpracováváme pomocí **výrazů**:

- Celá čísla počítáme pomocí **aritmetických operátorů (+, -, *, /, %)**.

```
3 int b = 1 + 1;       //b = 2 //scitani
4 int c = 2 - 3;       //c = -1 //odcitani
5 int d = b * c;        //d = -2 //nasobeni
6 int e = 5 / 2;        //e = 2 //celociselne deleni
7 int f = 5 % 2;        //f = 1 //zbytek po celociselnem deleni (modulo)
```

Pořadí aplikace operátorů se řídí jejich **prioritou**:

- Změnu pořadí můžeme vynutit pomocí kulatých závorek (**()**).

```
8 int g = 1 + 2 * 3;    //g = 7 //nasobeni ma vetsi prioritu nez scitani
9 int h = (1 + 2) * 3;  //h = 9 //nejvetsi prioritu maji zavoroky
```

Vstup a výstup provádíme funkcemi z knihovny `stdio.h`:

- Knihovny vkládáme na začátku souboru **direktivou** – `#include` –
- Vstupy načítáme funkcí `scanf`, výstup vypisujeme funkcí `printf`.

Prvním parametrem funkce `scanf` je formátovaný **textový řetězec** se značkou načítaného datového typu, druhým parametrem je **adresa** načítané proměnné:

- Celá čísla označujeme symbolem `(%i)`.
- **Adresu** proměnné získáme operátorem **reference** `(&)`.

```
1 int x; //vytvarime promennou jmenem "x"
2 scanf("%i", &x); //ze vstupu do ni nacitame hodnotu
```

Funkce `printf` vypisuje formátovaný řetězec, který značek může obsahovat více:

- Značky se při výpisu nahradí čísly, v pořadí ve kterém jsme je funkci předali.
- Řádky ukončujeme speciálním znakem **nového řádku** `(\n)`.

```
3 int y = 10; //vytvarime cele cislo "y" s hodnotou 10
4 int z = 20; //vytvarime cele cislo "z" s hodnotou 20
5 printf("%i + %i = %i\n", y, z, y+z); //vypisujeme "10 + 20 = 30"
```

Vyzkoušejte si:

- Vytvořte si dvě celá čísla (**výšku** a **šířku**) a ze vstupu do nich načtete hodnoty.
- Vypište **obsah** a **obvod** odpovídajícího **obdélníku**.

$$S = a * b$$

$$O = 2 * (a + b)$$

Například:

- (2, 3) => Obsah je 6
=> Obvod je 10
- (10, 20) => Obsah je 200
=> Obvod je 60

Podmínky

Podmínka je řídicí struktura rozhodující o provedení nějaké části kódu:

- Jazyk C zná tři typy podmínek (**if-else**, **switch-case** a **ternární operátor**).
- **Switch-case** a **ternární operátor** (?) si budete probírat pouze na přednáškách.

Podmínka typu **if-else** vždy začíná klíčovým slovem – **if** – (**pokud**):

- Následuje **hlavička** s pravdivostním výrazem a **tělo** s příkazy, které se mají provést pouze pokud podmínka **byla pravdivá**.
- Za prvním tělem může následovat slovo – **else** – (**jinak**) a **druhé tělo** s příkazy, které se mají provést pouze pokud podmínka **nebyla pravdivá**.

```
1 int x;           //vytvarime promennou jmenem "x"
2 scanf("%i", &x); //ze vstupu do ni nacistame hodnotu
3
4 if (x > 0)      //hlavicka podminky if
5 {              //zacatek tela if
6     printf("Cislo %i je kladne\n", x); //vypis pokud ano
7 }              //konec tela if
8 else           //hlavicka else
9 {              //zacatek tela else
10    printf("Cislo %i neni kladne\n", x); //vypis pokud ne
11 }              //konec tela else
```

Výsledkem pravdivostního výrazu je **pravdivostní hodnota** typu - `bool` - (`boolean`):

- Tento datový typ nabývá pouze hodnot - `true` - (**pravda**) a - `false` - (**nepravda**).
- Pravdivostní hodnoty získáváme použitím **relačních operátorů** (`<`, `<=`, `>`, `>=`, `==`, `!=`).
- Pozor na rozdíl mezi operátory porovnání (`==`) a přiřazení (`=`)!

```
1 if (x < 0)           //pokud je "x" mensi jak 0
2 if (x <= 0)          //pokud je "x" mensi nebo rovno 0
3 if (x > 0)           //pokud je "x" vetsi jak 0
4 if (x >= 0)          //pokud je "x" vetsi nebo rovno 0
5 if (x == 0)          //pokud je "x" rovno 0
6 if (x != 0)          //pokud "x" neni rovno 0
```

Pravdivostní hodnoty můžeme kombinovat **logickými operátory** (`&&`, `||`, `!`):

```
7 if (x>0 && y>0)     //pokud je "x" a zaroven "y" kladne (oba dva)
8 if (x>0 || y>0)     //pokud je "x" nebo "y" kladne (alespon jeden)
9 if (! x>0)           //pokud "x" neni kladne
```


Vyzkoušejte si:

- Ze vstupu načtete jedno celé číslo (X).
- Vypište jestli je X v rozsahu od 0 (včetně) do 10 (kromě).
- Vypište jestli je X sudé nebo liché (párne alebo nepárne).

Například:

- (5) => Cislo 5 v rozsahu je
=> Cislo 5 je liche
- (10) => Cislo 10 v rozsahu neni
=> Cislo 10 je sude

Pokud existují více jak dvě možnosti, podmínky můžeme **řetězit**:

- Za slovo – `else` – přidáme – `if` – s **další podmínkou**.
- Pokud je podmínka splněna více, vždy se provede **pouze ta první z nich!**
- Pokud **tělo** obsahuje jen jediný příkaz, složené závorky můžeme vynechat.

```

1  int x;                //vytvarime promennou jmenem "x"
2  scanf("%i", &x);    //ze vstupu do ni nacistame hodnotu
3
4  if (x > 0)           //pokud je "x" vetsi jak "0"
5  {
6      printf("%i je kladne\n", x);
7  }
8  else if (x < 0)     //jinak, pokud je "x" mensi jak "0"
9  {                   //(a "x" nebylo vetsi jak "0")
10     printf("%i je zaporne\n", x);
11 }
12 else                //jinak (pokud "x" nebylo ani vetsi ani mensi jak "0")
13     printf("%i neni ani kladne ani zaporne\n", x);
14                     //u posledniho else jsme vynechali zavorky

```

Složitější případy řešíme zanořováním:

- Do těla podmínky můžeme umístit celou další podmínku.

```
1 int x, y; //zraceny zapis pro
2 //int x; //vytvarime cele cislo "x"
3 //int y; //vytvarime cele cislo "y"
4 scanf("%i %i", &x, &y); //zkraceny zapis pro
5 //scanf("%i", &x); //do "x" nacistame hodnotu
6 //scanf("%i", &y); //do "y" nacistame hodnotu
7
8 if (y != 0) //pokud "y" neni nula
9 {
10     if (x % y == 0) //pokud je zbytek po deleni "x%y" roven "nule"
11         printf("%i je delitelem %i\n", y, x);
12     else //jinak (pokud zbytek po deleni nebyl roven nule)
13         printf("%i neni delitelem %i\n", y, x);
14 }
15 else //jinak (pokud "y" byla nula)
16 {
17     printf("Cislem %i nelze delit\n", y);
18 }
```

Vyzkoušejte si:

- Ze vstupu načtete jedno celé číslo (X) a vypište jeho **absolutní hodnotu**.
- Ze vstupu načtete tři celá čísla (A, B, C) a vypište jejich **maximum**.

Například:

- (5) => Absolutni hodnota je 5
- (-5) => Absolutni hodnota je 5
- (10, 20, 30) => Maximum je 30
- (10, 30, 20) => Maximum je 30
- (30, 20, 10) => Maximum je 30