Studentský matematicko-fyzikální časopis

M&M

Ročník XXIII / Číslo 2



*Časopis M&M a stejnojmenný korespondenční seminář je určen pro studenty středních škol, kteří se zajímají o matematiku, fyziku či informatiku. Během školního roku dostávají řešitelé zdarma čísla se zadáním úloh a témat k přemýšlení. Svá řešení odesílají k nám do redakce. My jejich příspěvky opravíme, obodujeme a pošleme zpět. Nejzajímavější řešení otiskujeme.*

**Milí čtenáři,**

je tu další číslo časopisu M&M. Najdete v něm zadání nových úloh a nových témat. Těšíme se na vaše řešení.

Podzimní soustředění je za námi, všichni studenti univerzity třetího věku zdárně přežili těžké životní situace, zázračně omládli a vrátili se do středoškolských lavic. Fotografie ze soustředění se brzy objeví na našem webu.

23. listopadu budou organizátoři M&M spolupracovat na Dni otevřených dveří na Matfyzu (kde bude i spousta jiných zajímavých věcí), a to v Kongresovém centru, nebo na Malé Straně v budově MFF UK.

Nejbližší seminářová akce je Vánoční víkendový sraz, který se bude konat 9.–11. prosince nedaleko Berouna. Rádi na něm uvidíme naše řešitele i jiné příznivce semináře M&M. Přihlášku a více informací najdete na našich internetových stránkách[1].

*Vaši organizátoři*

[1] `https://mam.mff.cuni.cz/soustredeni/pripravujeme/`

# Zadání úloh

**Termín odeslání druhé série: 6. 12. 2016**

*Crrr... Crrr... Sobota, 8:00. Luboš téměř ještě poslepu utišuje budík a přemýšlí, proč si ho vlastně včera večer nastavil. Vždyť je sobota! Jen co se probudí i druhá mozková buňka, Luboš vyskakuje z postele, rychle snídá, balí, a pospíchá na nádraží. Tam už ho netrpělivě vyhlížejí Verča s Tadeášem a Evelínou. Na poslední chvíli všichni společně naskakují do vlaku. Na příští zastávce přistupuje ještě Olda, sestava je kompletní. Dobrá nálada víří vzduchem, a tak cesta rychle ubíhá. Ferdinandov. Vystupovat! Verča dostala k narozeninám slevový voucher na novou akční hru Hranostaj, jejíž dějiště se nachází právě v místní opuštěné továrně. Čeká na ně 120 minut dřiny, potu a trápení mozkových závitů. Všichni se však těší. Alespoň zatím...*

*Již od začátku ale nedostanou nic zadarmo. Po rozehřátí při šplhu a troše balancování na kladině následuje místnost skrz naskrz prošitá ostnatým drátem. Kudy dál? Času není nazbyt, dveře na druhé straně místnosti se pomalu začínají zavírat...*

## Úloha 2.1 – Prásk (4b)

Spočítejte, za jak dlouho se zavřou obdélníkové dveře výšky $h$ o hmotnosti $m$, pokud jsou otevřeny o úhel $\alpha$, tlak na jedné straně dveří je $p_1$ a na druhé $p_2$. Neuvažujte velké proudění vzduchu.

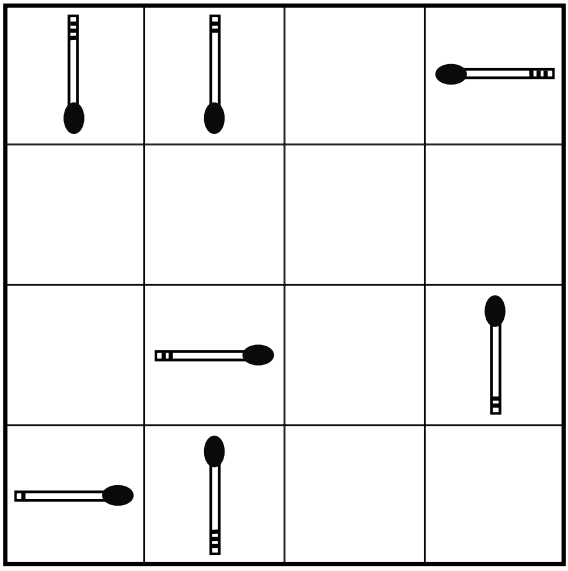
*Na poslední chvíli se uzoučkou škvírou ve dveřích prosoukává i Tadeáš. Stihli to! Jsou venku. Rychlý oddech a běží se dál. Blátivou stezkou se dostávají až k bažině, na jejímž okraji se nachází několik klád. Jak je umístit, aby mohli bezpečně přejít na druhou stranu?*

## Úloha 2.2 – Cesta po sirkách (3b)

Mějme čtvercovou síť o velikosti 4×4. Vezmeme si 16 sirek a uděláme na nich čárky následovně: na jednu sirku nakreslíme jednu čárku, na osm sirek dvě čárky, na sedm sirek tři čárky.

Položíme-li na políčko sirku s jednou čárkou, ukazuje ve směru hlavičky na sousední pole, sirka se dvěma čárkami o jedno pole dále a sirka se třemi čárkami na třetí políčko v daném směru. Podíváme se na pole, na které ukazuje sirka, a pokud na něm leží nějaká další, pokračujeme podle ní. Putování může skončit různě, nás však bude zajímat takové rozložení sirek, že se po navštívení poslední sirky vrátíme opět na začátek (viz Obr. 1).

Dokážete umístit všech 16 sirek na síť tak, aby tvořily uzavřenou cestu (viz Obr. 1)? Existuje jiné označení sirek, které umožní vytvořit delší uzavřenou cestu (za délku cesty se považuje počet políček)?



**Obrázek 1:** Ukázka cyklu na síti s použitím sedmi sirek (jedna s jednou čárkou, tři se dvěma čárkami a tři se třemi čárkami).

*Společnými silami překonávají bažinu s minimálními ztrátami – pouze Olda zde místním bohům obětoval čelenku. Ještě trocha bahna… Brzy však již následuje očista v ledové říčce. Přidávají do kroku, aby se trochu zahřáli. Navíc už jim také nezbývá moc času… Evelína svou výbornou střelou otevírá vstup do poslední části hry. Ještě trocha fyzického drilu a už se ocitají uprostřed velkého trojúhelníku nakresleného na zemi. Rychle obsazují vyznačené pozice. Pod jakým úhlem musí udeřit do velké černé koule, aby trefila červené tlačítko v jednom z rohů trojúhelníka?*

## Úloha 2.3 – Zapeklitý trojúhelník (3b)

Mějme rovnoramenný trojúhelník $ABC$. Nechť $M$ je střed jeho základny $AB$ a $N$ je bod osově souměrný s $M$ podle přímky $BC$. Rovnoběžka s $AB$ procházející bodem $N$ protíná přímku $AC$ v bodě $K$. Určete velikost úhlu $AKB$.

*Zvládli to! Do konce časového limitu zbývalo 23 vteřin! Vzájemně se chválí za výborné výkony, když se Tadeášův prázdný žaludek přihlásí o pozornost. Vydanou energii je třeba doplnit, a tak celá skupinka míří do blízké restaurace, kde znovu a znovu rozebírají, co všechno zažili. Ještě sladká tečka v cukrárně cestou na nádraží a vzhůru za teplem svých domovů. Cílová stanice se blíží, za okny se již rozprostírá tma. Ještě než se všichni rozloučí a vydají každý svým směrem, je potřeba vyrovnat účty.*

## Úloha 2.4 – Vyrovnání dluhů (3b)

Pět kamarádů spolu vyrazilo na akční hru Hranostaj. Verča zaplatila startovné, Tadeáš koupil lístek pro čtyři z nich do Ferdinandova, Olda si koupil svoji jízdenku a zaplatil také zákusky v cukrárně, Luboš se postaral o účet v restauraci. No a na Evelínu zbylo vymyslet, jak se co nejjednodušeji vyrovnat.

Obecněji máme $n$ kamarádů a $m$ dluhů, což jsou trojice (kdo, komu, kolik dluží). Nalezněte co nejlepší horní odhad na počet transakcí, který bude pro daná $n$ a $m$ na vyrovnání vždy stačit. Pak vymyslete co nejrychlejší algoritmus, který takové vyrovnání nalezne. Pokud dokážete, že úloha nalezení vyrovnání s *minimálním* počtem transakcí je NP-úplná[2], bonusové body vás neminou!

*Vyřešeno. Mějte se krásně a brzy na viděnou!*

# Zadání témat

## Téma 3 – Filmoví poradci

Vyberte si film a popište fyzikální, matematické nebo informatické nesmysly, které se v něm vykytují.

Navrhněte, jak tyto nesmysly odstranit, pokud to jde. V příspěvku popište část filmu, ve které se nesmyslná scéna odehrává (např. specifikací od a do které minuty filmu scéna probíhá).

[2] O NP-úplných problémech byla přednáška na soustředění. Jinak si můžete přečíst např. kuchařku KSP: `http://ksp.mff.cuni.cz/kucharky/tezke-problemy/`

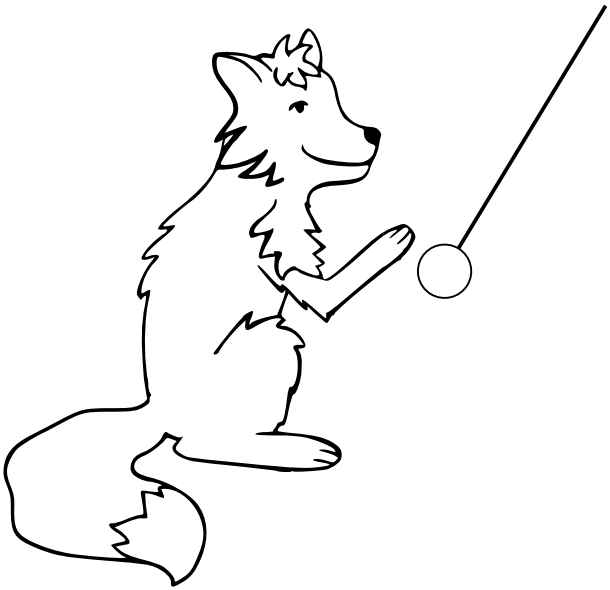
# Téma 4 – Simulace částic ve fyzice

Počítačové technologie rozšiřují možnosti řešení čím dál komplikovanějších fyzikálních úloh. Dříve byla dovednost umět využívat programování při fyzice výhodná, dnes je již prakticky nutná. Proto přinášíme téma věnované fyzikálním částicovým simulacím. Budete mít za úkol zkoumat metody simulací a porovnávat jejich vlastnosti. Nabídneme též pár krátkých návodů, které vám pomohou překonat bariéru k programování, pokud jste se s ním ještě vůbec nesetkali.

Vaším prvním úkolem je simulovat kmitavý pohyb harmonického oscilátoru pomocí několika metod a metody pak porovnat. V kmitavém pohybu je zrychlení $a$ popsáno rovnicí

$$a = -\omega^2 x,$$

jako funkce polohy $x$ při parametru $\omega$.



Pro úlohy tohoto typu (kde dynamiku nám určuje zadané zrychlení jakožto funkce polohy) je vždy potřeba znát počáteční podmínky, kterými je počáteční poloha a počáteční rychlost. Následně pak podle zvolené metody tyto hodnoty po krátkých časových krocích přepočítáváme.

Numerické metody většinou podléhají chybám (odchylkám od přesného řešení), které lze redukovat dělením na větší počet kratších časových úseků a nebo použitím jiné metody. Přestože již v dnešní době není problém provést simulaci s větším počtem kratších kroků, existují metody, u kterých zjemňování nepomáhá dostatečně, protože jejich chyba exponenciálně roste. Proto je snaha vyvíjet numerické metody tak, aby poskytovaly dostatečně přesné řešení i pro menší počty kroků.

Všechny metody fungují tak, že v každém kroku „staré“ hodnoty nahradí „novými“ podle určitého schématu. Poté hodnoty, které jsou nové, použijí jako staré pro příští krok. Pro výpočet nových hodnot na základě starých použijte dvě různé metody:

**Metoda A**

$$
\begin{aligned}
t_{\mathrm{new}} &\leftarrow t_{\mathrm{old}} + h \\
x_{\mathrm{new}} &\leftarrow x_{\mathrm{old}} + h \cdot v_{\mathrm{old}} \\
v_{\mathrm{new}} &\leftarrow v_{\mathrm{old}} + h \cdot a(x_{\mathrm{old}}) \\
x_{\mathrm{old}} &\leftarrow x_{\mathrm{new}} \\
v_{\mathrm{old}} &\leftarrow v_{\mathrm{new}} \\
t_{\mathrm{old}} &\leftarrow t_{\mathrm{new}}
\end{aligned}
$$

**Metoda B**

$$
\begin{aligned}
t_{\mathrm{new}} &\leftarrow t_{\mathrm{old}} + h \\
x_{\mathrm{new}} &\leftarrow x_{\mathrm{old}} + h \cdot v_{\mathrm{old}} \\
v_{\mathrm{new}} &\leftarrow v_{\mathrm{old}} + h \cdot a(x_{\mathrm{new}}) \\
x_{\mathrm{old}} &\leftarrow x_{\mathrm{new}} \\
v_{\mathrm{old}} &\leftarrow v_{\mathrm{new}} \\
t_{\mathrm{old}} &\leftarrow t_{\mathrm{new}}
\end{aligned}
$$

Hned v prvním příkladu si tedy vyzkoušejte nasimulovat průběh pohybu harmonického oscilátoru. Jako počáteční podmínky volte polohu a rychlost

$$
\begin{aligned}
x &= 1\,\mathrm{m} \\
v &= 0\,\mathrm{m/s},
\end{aligned}
$$

přičemž zrychlení je popsáno výše uvedenou rovnicí, kde volte $\omega = 1\,\mathrm{s}^{-1}$.

Na konci každého kroku je vhodné hodnotu $x_{\mathrm{new}}$ uložit do souboru pro následnou analýzu v grafickém programu[3] a dále pokračovat opakováním cyklu. Je vhodné zobrazovat závislost $x$ (poloha) na $t$ (čas). Pro začátek doporučujeme nastavit hodnotu časového kroku $h = 0{,}1\,\mathrm{s}$ a počet kroků simulace na 200.

Obě metody budou zatíženy určitou chybou oproti přesnému řešení, kterým je funkce $x = \cos(\omega t)$. Chyby budou různě velké, ale velmi důležité je též to, jakým způsobem se chyba projevuje. Pro tuto analýzu zkoušejte nastavit i jinak veliké časové kroky a měňte jejich počet. Pokud si troufáte, můžete se pokusit vysvětlit, proč jsou tyto metody schopny se přiblížit k přesnému řešení pro dostatečně malý krok $h$.

Zkuste vlastními slovy popsat, v čem se výsledky těchto dvou metod liší. Zamyslete se především nad následujícími aspekty simulace:

- Všímáte si chyby, která se postupně s průběhem simulace nasčítává exponenciálně? Tedy že každá odchylka podporuje nárůst odchylky v následujících krocích, a výsledná chyba tak roste exponenciálně s počtem kroků simulace.

- Všímáte si chyby, která se postupně s průběhem simulace nasčítává lineárně? Tedy že v každém kroku dojde k odchylce, která je nezávislá na ostatních, a výsledná chyba je pak přímo úměrná počtu kroků simulace.

[3] Při velkém počtu kroků simulace je vhodnější ukládání do souboru provádět až po delším časovém intervalu. Simulace se tím výrazně zrychlí.

- Všímáte si chyby, která postupně s průběhem simulace osciluje? Tedy že chyby v jednotlivých krocích se mezi sebou navzájem odečtou, a pro libovolně dlouhou simulaci se tak chyba nenasčítá.

Nyní si povíme pár rad, jak technicky zvládnout toto téma. Ukážeme si základní program v jazyce C++, pomocí kterého je možné provést základní simulaci a uložit si výsledky do souboru. Doporučíme programy, s jejichž pomocí vše pohodlně zvládnete. Volíme programy a postupy, které jsou hojně využívány ve vědě, a pokud plánujete studovat na MFF UK, tak se s některými z nich dříve nebo později setkáte. Přesto můžete použít jakékoli jiné nástroje a programovací jazyky, zejména pokud je již ovládáte.

Další doporučenou možností je použít jazyk Python. V 17. ročníku M&M o něm vyšel seriál. Na adrese `https://mam.mff.cuni.cz/rocnik/17/` si můžete stáhnout čísla s jednotlivými díly seriálu.

Pro začátek bude potřeba si nainstalovat editor, kde budeme psát náš program. Doporučujeme program Dev-C++, který naleznete na webu `http://orwelldevcpp.blogspot.cz/`. Editor v sobě zahrnuje aktuální verzi kompilátoru TDM-GCC, který bude váš kód překládat do instrukcí pro váš počítač. Pro svůj první program stačí pouze přes cestu `File -> New -> Source File` vytvořit váš první soubor s kódem programu. Hned si vyzkoušejte první jednoduchý kousek programu. Do nového souboru vložte následující text:

```
#include <cstdio>

int main()
{
    int i;
    double a = 2.0, x = a;
    FILE* DATA = fopen("data.txt", "w");

    for (i = 1; i <= 6; i = i+1) {
        x = (x + a / x) / 2.0;
        fprintf(DATA, "%d %.15f\n", i, x);
    }

    fclose(DATA);
    return 0;
}
```

Pro spuštění programu pak stačí volit `Execute -> Compile & Run`. Pokud jste tak ještě neučinili, tak vás Dev-C++ vyzve k tomu, abyste zdrojový kód vašeho programu někam uložili na disk.

Jedná se o krátký kus kódu, který v sobě koncentruje skoro vše, co bude potřeba znát pro psaní vlastních simulací. Vysvětlíme si, co přesně provede.

První řádek, `#include <cstdio>`, zpřístupní k dalšímu použití v programu příkazy pro práci se soubory.

Následuje funkce `int main() { ... return 0; }`, do které místo `...` vepíšeme, co má náš program vykonat. V našem příkladu následuje příkaz `int i`. Tímto příkazem programu oznamujeme, že hodláme používat proměnnou, kterou nazýváme `i` a výrazem `int` dáváme najevo, že se jedná o datový typ `integer` – nejběžnější typ pro ukládání celých čísel. Všechny proměnné musí být tímto způsobem deklarovány, aby bylo možné s nimi dále pracovat.

Na dalším řádku `double a = 2.0, x = a;` zakládáme další dvě proměnné, tentokrát typu `double`, což je jeden z typů reprezentujících reálné číslo. Tentokrát za názvy proměnných přidáváme i počáteční hodnoty, aby proměnná již na začátku danou hodnotu obsahovala. Je dobré vědět, že symbol `=` v C++ znamená přiřazení (tedy ← z obecného popisu průběhu simulace). Pokud chceme rovnost v matematickém smyslu slova, je potřeba použít `==`. Do proměnné `x` se v našem kódu uloží hodnota, která je v proměnné `a`, tedy číslo 2. Mějte na paměti, že reálné číslo zapsané přímo do kódu v sobě musí obsahovat desetinnou tečku, takže v případě čísla 2 musíme volit zápis 2.0 [4]. Kompilátor sice v mnoha případech i bez toho zápis správně interpretuje jako operaci s desetinnými čísly, ale přesto velmi doporučujeme ji nevynechávat. Všimněte si dále, že na konci každého příkazu máme středník – tímto ukončujeme jednotlivé příkazy. V příkazu

[4] V jazyce C++ se desetinná čísla píšou ve formátu s desetinnou tečkou, proto budeme tento termín v textu dále používat, přestože se správně v češtině používá desetinná čárka.

`double a = 2.0, x = a;` máme `a` a `x` odděleno pouze čárkou, čímž naznačujeme, že ještě stále definujeme proměnné typu `double`. Až středníkem ukončíme tuto deklaraci.

Dále následuje příkaz `FILE* DATA = fopen("data.txt","w");`. V tomto příkazu vytváříme `FILE*`, což je odkaz na soubor. Soubor jako takový bude uložen v počítači pod názvem `data.txt` ve stejné složce, kde spouštíte váš kód. Příkaz `fopen` s parametrem `w` znamená, že program vytvoří soubor v počítači, pokud ještě neexistuje, a má dovoleno do něj zapisovat. Pokud soubor existuje a má nějaký obsah, tak program jeho obsah smaže. Tímto jsme našemu programu oznámili, s jakými proměnnými bude pracovat a že si má připravit soubor, kde bude své výstupy ukládat.

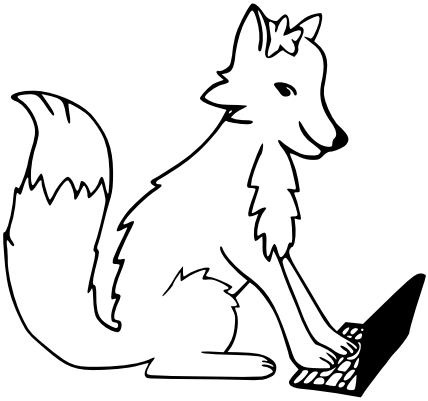
Následuje cyklus

```
for (i = 1; i <= 6; i = i+1) {
      ...
}
```

Zde oznamujeme, že proměnná `i` nám bude kontrolovat průběh cyklu. Na počátku zde uložíme hodnotu 1 a dokud je hodnota `i` menší nebo rovna šesti, provede se vnitřek cyklu, a poté `i` navýšíme o 1. Tím jsme zajistili, že náš cyklus proběhne právě šestkrát. Uvnitř cyklu místo `...` umístíme příkazy, které se provedou opakovaně (v našem příkladu šestkrát). Podle prvního řádku `x = (x + a / x) / 2.0;` dojde k přepsání hodnoty `x` novou hodnotou, která se spočte z výrazu za symbolem `=`.

Podle druhého řádku `fprintf(DATA, "%d %.15f\n", i, x);` náš program zapíše do souboru, na který odkazuje proměnná `DATA` (jedná se o soubor `data.txt`, jak jsme deklarovali na začátku). Značky začínající procentem (`%d`, `%.15f`) uvnitř příkazu mají speciální význam – budou nahrazeny dalšími parametry příkazu (`i` a `x`). Tedy, nejdříve bude vypsáno celé číslo (tomu odpovídá `%d`), následně mezera a po ní reálné číslo (`double`) s celkem 15 číslicemi za desetinnou tečkou (tomu odpovídá `%.15f`). Pak následují parametry `i` a `x`, které říkají, že první tag `%d` bude nahrazen hodnotou uloženou v `i` a druhý tag `%.15f` bude nahrazen hodnotou uloženou v `x`. `\n` znamená nový řádek v souboru. Doporučujeme pak po proběhnutí programu otevřít soubor `data.txt` a podívat se, co nám přesně vzniklo. Pokud jste vše provedli správně, mělo by se v souboru `data.txt` objevit

```
1 1.500000000000000
2 1.416666666666667
3 1.414215686274510
4 1.414213562374690
5 1.414213562373095
6 1.414213562373095
```



Přejete-li si data vykreslit, doporučujeme k tomu požít nástroj GNUPLOT, který stáhnete zde `http://gnuplot.info/`[5]. Program stačí jen stáhnout a spustit. Pro vykreslení grafu pak stačí zadat v programu příkaz

```
plot 'C:\vaseslozka\data.txt' with lines
```

a potvrdit enterem. Místo `C:\vaseslozka` zadejte vámi zvolenou adresu souboru. Na konci příkazu užíváme postfix `with lines`, což znamená, že jednotlivé body grafu budou spojeny čarou. Pokud si to nepřejete, tak tato dvě slova jednoduše vynechejte. Gnuplot pak vezme soubor `data.txt` a vykreslí dvojrozměrný graf. První sloupeček vždy bere jako $x$-ové souřadnice vykreslovaných bodů, druhý jako $y$-ové.

Pokud jste vše zvládli, tak jste napsali svůj první kus kódu, který umí velmi efektivně počítat odmocninu z reálného čísla. Program vždy spočítá odmocninu z čísla $a$, které zadáte na začátku při deklaraci proměnných.

Na tomto příkladu vidíme účinnost použité metody. Všimněte si, že se počet desetinných míst, která odpovídají tabulkové hodnotě odmocniny ze dvou, s každým krokem zdvojnásobuje. Stačí tedy jen pár kroků na výpočet s velmi vysokou přesností.

Ukázkový kód jsme se snažili alespoň rámcově vysvětlit. Nyní byste měli mít přehled o tom, co dělá který řádek kódu. Popsat přesný způsob, jakým se kód vykonává a jak to celé uvnitř funguje, by vyžadovalo mnohem delší povídání. Zájemcům doporučujeme k přečtení seriál *C++ pod lupou* od Vladimíra Klimovského[6]. Pokud byste se při řešení tématu potýkali s technickými problémy a nevěděli si rady, určitě nám dejte vědět, rádi vám poradíme.

*Mára*

[5] Podrobnější informace o GNUPLOTu můžete najít v jednom z našich starších čísel na adrese `https://mam.mff.cuni.cz/media/cislo/pdf/16/16-4.pdf`

[6] K dispozici na `http://www.pcrevue.sk/a/C---pod-lupou---Prva-cast--Uvod`


## Adresa redakce:

M&M, OVVP, MFF UK
Ke Karlovu 3
121 16 Praha 2

*E-mail:* mam@matfyz.cz
*WWW:* http://mam.matfyz.cz