ZÁPADOČESKÁ UNIVERZITA V PLZNI
FAKULTA PEDAGOGICKÁ
KATEDRA VÝPOČETNÍ A DIDAKTICKÉ TECHNIKY

KOMUNIKATIVNÍ PŘÍSTUP K VÝUCE PHP

DIPLOMOVÁ PRÁCE

Jan Pejša
učitelství pro střední školy, M–VT
(1998–2004)

Vedoucí diplomové práce: Ing. Václav Vrbík, CSc.
Plzeň, leden 2004

Prohlašuji, že jsem diplomovou práci vypracoval samostatně s použitím uvedené literatury a zdrojů informací.

V Plzni dne 5. ledna 2004

Jan Pejša

Na tomto místě bych chtěl poděkovat vedoucímu diplomové práce Ing. Václavu Vrbíkovi, CSc. za zájem, připomínky a čas, který věnoval mé práci. Mé poděkování patří rovněž všem, kteří mi s prací jakkoliv pomohli.

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Cílem diplomové práce je analyzovat možnosti využití prvků komunikativního přístupu k učení známého z výuky cizích jazyků. Při výuce PHP skriptu navrhuji ilustraci vybraných postupů při samotné výuce. Diplomová práce je členěna do jednotlivých kapitol.

Kapitola 2 se zabývá možnostmi využití prvků komunikativního přístupu k učení známého z výuky cizích jazyků při výuce PHP skriptu na střední škole. Kapitola 2.2 shromažďuje postupy využitelné při výuce programování se zaměřením na skriptovací jazyk PHP (vycházíme z výuky cizích jazyků — viz. kapitola 2.1). Následující kapitola 3 ukazuje praktické využití komunikativních metod při výuce PHP skriptu. Tato kapitola je silně provázána s pracovní aplikací, která je popsána v kapitole 4. Ta pomáhá při řešení úloh uvedených v diplomové práci a jsou v ní také všechny skripty, které se v kapitole 3 vyskytují. Využití je velmi výhodné, jelikož žák v levé části vidí zdojový kód PHP skriptu a v pravé části výsledné zpracování skriptu, které je okamžité. Pracovní aplikace je dostupná na internetu[1] a také na přiloženém CD–ROM. Diplomová práce se nezabývá popisem skriptovacího jazyka PHP, i když je často v některých pasážích vysvětlován. Diplomová práce vyžaduje základní znalosti značkovacího jazyka HTML a znalosti jakéhokoliv strukturovaně pojatého programovacího jazyka.

Součástí práce jsou také přílohy. V příloze B respektive A jsou uvedeny zdrojové kódy pracovní aplikace respektive stránek diplomové práce. Další přílohou je CD–ROM, kde mimo jiné naleznete tuto práci v elektronické podobě (formát PDF).

[1] Umístěno na serveru určeném pro domácí stránky studentů Západočeské univerzity, <http://home.zcu.cz/~jpejsa/diplomka/>.

Kapitola 2

Komunikativní přístup ve výuce programování

2.1 Komunikativní přístup ve výuce cizích jazyků

2.1.1 Historie komunikativního přístupu k výuce

Komunikativní výuka jazyků se zabývá hledáním a užitím metod použitelných ve vyučování. Základy komunikativní výuky jazyků mají kořeny již ve třicátých letech dvacátého století, avšak současné formy komunikativní výuky jazyků sahají až do pozdních šedesátý let, kdy docházelo v Británii ke značným změnám v tradičním vyučování cizím jazykům. Do té doby byl hlavní přístup k učení pomocí situačního jazykového vyučování. V situačním vyučování jazyků byl jazyk vyučován na základě cvičení základních struktur ve smysluplných aktivitách, založených na situacích. Dále k rozvoji komunikativního přístupu k učení přispěla vzrůstající potřeba komunikace na mezinárodní úrovni, zejména v evropských zemích. Komunikativní přístup k učení se rychle rozšířil i mimo anglicky mluvící země, přestože byl původně zamýšlen pro inovaci nevyhovujících osnov výuky angličtiny ve Velké Británii. Postupně se stal jedním z nejpoužívanějších a nejvýznamnějších přístupů k výuce jazyků.

U nás se situace změnila až po roce 1989, kdy začala vyšší poptávka po kvalitních znalostech západoevropských jazyků. Většina nových učebnic zaměřená na výuku cizích jazyků je pojata komunikativním přístupem (4; 14). Pokud se však zaměříme na výuku programovacích jazyků, najdeme komunikativní přístup v minimálním počtu učebnic. Navíc vždy jen ve formě jednotlivých úkolů nebo cvičení.

Komunikativní přístup k učení programovacího jazyka je nová záležitost, která má umožnit efektivnější vyučování programovacího jazyka na školách.

2.1.2 Prvky komunikativní výuky

Komunikativní přístup k učení je velmi komplexní systém zabývající se nejen obsahem výuky, ale také metodami a celým procesem výuky. Velký význam má i přiblížení výuky reálnému životu. V tabulce 2.1 je uvedeno porovnání mezi audio–linguální a komunikativní výukou jazyků (částečně převzato a upraveno z (3)). Tabulka ukazuje některé z hlavních rozdílů mezi komunikativními přístupy a dřívějšími tradicemi v jazykovém vyučování. Široce pojatý komunikativní přístup a relativně rozmanité způsoby, jakými je interpretován a aplikován, může být přisouzen faktu, že praktici z různých vzdělávacích tradic se s ním mohou identifikovat a zároveň ho mohou interpretovat rozdílně. Společná je ale teorie jazykového vyučování, která vychází z komunikativního modelu jazyka. Využití jazyka se pokouší převést do návrhu pro výukový systém, materiálů, role učitele a chování učících se, třídní aktivity a techniky.

2.1.3 Lingvistické přístupy, teorie

Komunikativní přístup ve vyučování jazykům stojí na bázi teorie jazyka jako komunikace (prostředku komunikace). Cílem vyučování jazyka je rozvoj schopnosti komunikovat. Osoba, která získá „schopnost komunikovat", získá jak znalosti, tak schopnost užívat jazyk s ohledem na to:

- zda (a do jaké míry) je něco formálně přípustné,
- zda (a do jaké míry) je něco (určitý jazykový jev) proveditelné, realizovatelné,
- zda (a do jaké míry) je něco vhodné, adekvátní vzhledem ke kontextu,
- zda (a do jaké míry) je něco už provedeno a co to znamená.

Jiná lingvistická teorie komunikace upřednostňovaná v komunikativní výuce cizího jazyka je funkčně pojaté užívání jazyka. Jazykověda je soustředěna na popis mluvních aktů nebo textů, neboť pouze studiem forem užití jazyka jsou v centru pozornosti všechny jeho funkce a významy. Tato teorie doplňuje předchozí. Popisuje sedm základních funkcí, které jazyk nabízí dětem, užívajícím ho jako mateřský:

1. instrumentální funkce (užití řeči — konání, čin, výkon),
2. regulatorní funkce (kontrola chování druhých),
3. interakční funkce (interakce s ostatními),

Tabulka 2.1: Porovnání audio–linguální a komunikativní výuky jazyků.

Audio–linguální výuka jazyků:	**Komunikativní výuka jazyků:**
Dbá o strukturu a formu více než o význam.	Význam považuje za nejdůležitější.
Požaduje memorování dialogů založených na strukturách.	Jsou–li použity dialogy, jsou soustředěny kolem komunikativních funkcí a nejsou normálně memorovány.
Jazykové jednotky nemusí být kontextualizovány.	Kontextualizace je základním předpokladem.
Učení se jazyku je učení se strukturám, zvukům nebo slovům.	Učení se jazyku je učení se komunikaci.
Je požadovaná rutina nebo přeučení se.	Je požadovaná efektivní komunikace.
Drilování je centrální technikou.	Drilování se může vyskytovat pouze okrajově.
Je požadovaná výslovnost podobná výslovnosti rodilého mluvčího.	Je požadovaná přiměřeně srozumitelná výslovnost.
Komunikativní aktivity přicházejí až po dlouhém procesu drilování a cvičení.	Pokusy komunikovat mohou být podporovány hned na začátku.
Je zakázáno použití mateřského jazyka.	Uvážlivé použití mateřského jazyka je uznáno ve vhodných příležitostech.
V počátečních stádiích je zakázán překlad.	Překlad se může použít, když ho studenti potřebují nebo z něj mají užitek.
Dokud není zvládnuto mluvení, tak je odloženo čtení a psaní.	Čtení a psaní mohou začít od prvního dne, je-li žádáno.
Cílový jazykový systém bude určen skrz zjevné vyučování vzorů systému.	Cílový jazykový systém bude nejlépe naučen skrz snahu o komunikování.
Druhy jazyka jsou uznány, ale ne vyzdviženy.	Jazyková různorodost je centrálním pojmem v materiálech a metodologii.
Řazení jazykových jednotek je určeno pouze principy jazykové složitosti.	Řazení je určeno zvážením obsahu, funkce nebo významu, který udržuje zájem.
Učitel řídí žáky a brání jim dělat cokoliv, co neodpovídá teorii.	Učitelé pomáhají studujícím jakýmkoli způsobem, který je motivuje k práci s jazykem.
Jazyk je zvyk, takže žák se musí za každou cenu vyhýbat chybám.	Jazyk je tvořen jedincem, často pokusem a omylem.
Přesnost je, v pojetí formální správnosti, hlavním cílem.	Plynulost a přijatelný jazyk jsou základními cíli, přesnost je souzena v kontextu.
Studenti mají spolupracovat s jazykovým systémem, ztělesněnými stroji nebo řízenými materiály.	Studenti mají vzájemně navazovat kontakt s ostatními lidmi a to buď přímo, při párové nebo skupinové práci nebo písemně.
Vnitřní motivace vznikne ze zájmu o strukturu jazyka.	Vnitřní motivace vznikne ze zájmu o to, co se jazykem sděluje.

4. osobní funkce (vyjadřování pocitů a mínění),
5. heuristická funkce (učení a objevování),
6. imaginativní funkce (svět fantazie),
7. reprezentativní funkce (informativní užití).

Učení cizího jazyka bylo podobně posuzováno příznivci komunikativní výuky jazyků jako získávání jazykových prostředků umožňující užívat různé funkce jazyka. Často citovaný teoretik je Henry Widdowson. Ve své knize „Učení jazyku jako komunikace" (14) předkládá pohled na vztah mezi jazykovými systémy a jejich komunikační hodnotou v textu a rozhovoru. Zaměřil se na komunikační akty, při nichž je nutné užívat jazyk k různým účelům.

Další analýzu jazykové kompetence nabízí Canale a Swain (4), kteří definují čtyři dimenze komunikativní kompetence:

- gramatická (lingvistická, gramatika a slovíčka),
- sociolingvistická (porozumění jazyku ve společenském kontextu a role příbuzenství),
- diskuzní (interpretace zprávy),
- strategická (strategie, jak akt komunikace začít, pokračovat v něm, přerušit ho, obnovit, skončit).

2.1.4 Cíle komunikativní výuky

Principy pro komunikaci:

1. princip integrující (jazyk jako prostředek vyjádření),
2. princip jazykový (jazyk jako systém a objekt učení),
3. princip afektivní a princip mezilidských vztahů (jazyk jako prostředek ocenění ostatních),
4. princip individuálních učebních potřeb (učení založené na analýze chyb),
5. obecný vzdělávací princip (učení cizího jazyka podle školního kurikula).

Tyto principy jsou obecné cíle, použitelné pro libovolnou situaci ve výuce. Cíle specifické pro komunikativní výuku jazyků nemohou být definovány bez definování těchto obecných cílů, předpokládáme-li, že výuka cizích jazyků bude

respektovat individuální potřeby každého žáka. Tyto potřeby se projevují zejména v oblastech psaní, čtení, poslechu nebo mluvení. Kurikulum nebo osnovy pro určitý kurs (předmět) musí zohledňovat úroveň znalostí žáka a jeho komunikativní potřeby.

2.1.5 Osnovy

V centru komunikativní výuky jazyků stojí diskuse o povaze učebních osnov. Existuje jistý teoretický model, který specifikuje významově–gramatické kategorie (např. frekvenci jazykových výrazů, jejich pohyb a umístění) a kategorie komunikativních funkcí, umožňující studentovo vyjádření. Tento model byl rozšířen do učebních osnov, které popisují cíle, jichž by při výuce cizího jazyka měl student dosáhnout. Dále situace, ve kterých student cizího jazyka použije (cestování, obchod), témata, o kterých bude mluvit (popis osoby, vzdělání, nakupování), činnosti, při kterých jazyk použije (popis, zprostředkování informací, vyjádření souhlasu nebo nesouhlasu), pojmy užívané při komunikaci (čas, frekvence, trvání), stejně jako potřebná slovíčka a gramatiku.

Diskuse o tom, jak má vypadat učební plán, jsou v komunikativní výuce jazyků velmi rozsáhlé. Originální teoretický model byl velmi brzy kritizován britskými jazykovědci jako model, který pouze nahrazuje jeden seznam požadavků (seznam gramatických jevů) seznamem jiným (seznam pojmů a funkcí). Tedy, že specifikuje produkty, ne procesy komunikace. Uvádí se, že teoreticko-funkční kategorie poskytují pouze částečný a nepřesný popis sémantických a věcných pravidel, kterých člověk při komunikaci s ostatními využívá. Neříkají nám nic o myšlenkových procesech nutných k tomu, aby člověk při komunikaci s ostatními mohl tato pravidla využít. Pokud ale chceme přijmout komunikativní výuku jazyků jako výuku, jejímž hlavním účelem je rozvoj komunikačních schopností, musí stát řeč v centru naší pozornosti.

2.1.6 Typy aktivit pro učení a výuku

Množství typů cvičení a aktivit je neomezené. Rozlišuje mezi „funkčními komunikačními aktivitami" a „sociálními interakčními aktivitami" jako základními typy v komunikativní výuce jazyků. Funkční komunikační aktivity zahrnují taková cvičení jako: hledat shody a rozdíly mezi obrázky, objevit chybějící objekt v mapě nebo obrázku, jeden žák popisuje druhému, jak nakreslit obrázek apod. Sociální interakční aktivity obsahují konverzaci, diskusi, dialogy, hry rolí, simulace, scénky, improvizaci a debaty.

- *Lingvistické základy:*
  Výsledky strukturalismu se zde rozšiřují o výsledky kontextualismu. Ve středu pozornosti už nestojí bezchybný jazykový projev. Všímáme si také účelu, dojmu, průběhu komunikace a situace, ve které se odehrává. Důraz je tedy kladen na adekvátní porozumění a situaci.

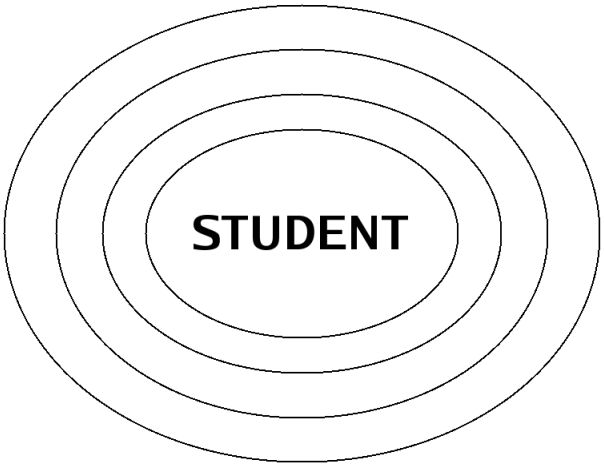
- *Teorie:*
  Jazykové počínání je zde viděno jako intelektuální a kreativní činnost. Objem a kvalita jazykového projevu ve vyučování závisí na tom, jak dalece žák porozumí cizojazyčným textům a v jaké míře ovládá patřičné jazykové prostředky. Postup podmiňování ve smyslu teorie behaviorismu je akceptován nejvýše v počáteční fázi výuky, jinak je přístup k němu kritický.

- *Literatura a texty:* zde rozlišujeme:
  „pedagogicko–funkcionální aspekt", tzn. že důraz je kladen na autentické texty; význam porozumění celku a selektivního porozumění stojí v popředí — žákům je tedy nabízen odpovídající postup a látka;
  „pedagogický aspekt", tzn. zde je ve vyučování důraz kladen na literární texty; jejich úlohou je přispívat k vývoji identity žáka.

- *Pedagogická teorie:*
  Žák rozvíjí svou vlastní osobnost a učí se artikulovat svá přání či záměry a prosazovat se v poměru k ostatním (tvořit vlastní identitu). Žák by se měl naučit pochopit svou vlastní roli a popřípadě ji odmítnout. Žák se učí schopnosti vcítit se do postavení druhých (empatie). V průběhu učebního procesu by měl žák dojít k poznání a naučit se akceptovat, že jím zastávané stanovisko není jediná správná pravda.

- *Didaktické a metodické principy:*
  a) Důraz spočívá na volbě témat: Probíraná témata by měla žáka nejen zajímat, ale také se ho přímo dotýkat. Žák by tak měl být motivován k učení se jazyku. Do vyučování jsou integrována také ostatní média.
  b) Důraz spočívá na jednání: Vyučování si všímá pragmaticko–funkcionálních kategorií a obsahuje práci s autentickými texty. Pedagogickým prostředkem je zde třeba trénink žáka v různých rolích, simulace a plánované hry. Frontální výuka je ve velké míře nahrazena prací ve skupinách. Jednotlivé schopnosti nevystupují izolovaně, ale ve spojení s ostatními.

Obrázek 2.1: Role žáka — vyjádření pomocí soustředných kruhů



2.1.7 Role žáka

V komunikativní výuce jazyků najdeme díky její koncepci širší spektrum rolí vymezených žákovi než v tradičních teoriích. Při komunikativní výuce jazyků chybí často text, nejsou vykládána gramatická pravidla, uspořádání učebny není standardní, studenti kooperují mezi sebou za podpory učitele, chyby nejsou opravovány, nebo ne vždy. Kooperace je upřednostňována před individualismem. Metodologové komunikativní výuky jazyků upozorňují na to, že se žáci učí poznávat, že porucha komunikace není chybou pouze mluvčího nebo posluchače, ale chybou obou stejně. Podobně úspěšná komunikace je výsledkem snažení všech zúčastněných stran.

Situace, ve které se žák nachází, se dá vhodně a přesně vyjádřit rozdělením na čtyři soustředné kruhy se studentem uprostřed (částečně převzato podle Billowse (2)).

1. První, těsně okolo žáka ležící vnitřní kruh, obsahuje všechno, co se dá bezprostředně vidět, slyšet nebo vzít do ruky. V praxi to znamená místnost, ve které se vyučuje. K tomu patří také okolí, které je ze třídy lehce vidět a slyšet — kupříkladu oknem. Patří sem také exkurze, které třída pod vedením učitele podniká. Zde se nabízejí možnosti vyučovací látku a slovní zásobu opakovat. Zároveň se tu vyskytuje možnost do naučených jazykových struktur přibírat nová slova, která zvolené okolí nabízí.

2. Druhý kruh, jehož dosažení by se mělo dít jenom po absolvování (a za pomoci) kruhu prvního, obsahuje všechno, co žák na základě vlastních zkušeností zná. Jsou to data a informace, které žák už viděl nebo je poznal v jiných situacích — i když momentálně není žádná z těchto věcí

přítomna. Učiteli musí být jasné, že vyvolat žákům i známé činnosti, objekty a situace tohoto kruhu ve sféře jejich vědomí je namáhavé, protože je v daném okamžiku není ani vidět ani slyšet. Zásadně tedy bychom se této problematice měli blížit jenom za pomoci situací prožívaných přímo ve třídě.

3. Ve třetím kruhu se setkáváme se vším, s čím žák nemá přímé zkušenosti, co se však za použití své fantazie a za pomoci různých plánů, map, dramatických scén, obrazů, modelů a jiných vizuálních pomůcek dokáže představit. Každá práce uvnitř tohoto třetího kruhu se opírá o ukončenou práci ve dvou prvních kruzích: neznámé věci dokážeme fantazií obsáhnout jen prostřednictvím věcí známých. Učitel si tu musí při probírání nové látky pomáhat všemi možnými prostředky, dramatickými gesty nebo kreslením na tabuli.

4. Čtvrtým kruhem je rozuměno všechno, co je žákovi představeno prostřednictvím mluveného, psaného nebo tištěného slova, bez podpory audiovizuálních prostředků. Patří sem tedy všechno, co náš žák v budoucnu v jazyce, který se učí, uslyší nebo si osvojí. Během učení se ale musí jeho jazykové schopnosti ve sféře předcházejících kruhů náležitě upevnit. Žák musí v těchto sférách cizí řeči dostatečně často naslouchat, rozumět a také umět používat jazyk, který mu otevře jiný svět.

2.1.8 Role učitele

Učitel má dvě základní role. První je napomáhat komunikaci mezi všemi zúčastněnými aktéry, různými aktivitami a texty. Druhá role je vystupovat jako nezúčastněný pozorovatel učební skupiny. Druhá role úzce souvisí s první a vychází z ní. Z těchto rolí plynou role další, sekundární: analytik potřeb, rádce a manažer skupinových procesů. Popíšeme si je.

Vyučující si nesmí nechat vzít možnost vyučování řídit. Jenom tehdy je schopen kontroly a jenom tehdy je schopen případné nutné modifikace nebo změny struktury učební látky. Pouze za těchto okolností si může být jistý tím, že se žáci učí výhradně správným formám. Vyučující pomáhá, když se vynoří nějaké problémy a v jednotlivých případech se stará i o další učební materiál, když vidí, že ten či onen žák má s látkou v učebnici potíže.

Jedním z nejdůležitějších poslání učitele je, aby dal vždy znova svým žákům příležitost slyšet naučená slova v kontextu situace. To znamená používání výrazných významových kontextů, aby slovům umožnily zformovat v mysli žáka podobu, a tím po sobě zanechat trvalé stopy, které vyvolávají asociace,

jakmile se opakují. Když vyučující zpozoruje, že se takový proces dal do pohybu, měl by dát svým žákům příležitost k formování jednotlivých útvarů a nechat je, aby pomocí nového prostředku prozkoumali současnou situaci sami. Žáci mají jazyku nejen rozumět, ale mají ho hlavně umět použít v různorodých situacích.

2.1.8.1 Analytik potřeb

Učitel přejímá odpovědnost za žákovy individuální potřeby při výuce cizích jazyků. Současný stav zjistí při individuálním rozhovoru se studentem, kdy s ním mluví o jeho učebním stylu, návycích a úspěších. Existují také formální testy, s jejichž pomocí učitel zjistí potřebné informace. Můžeme uvést několik důvodů, proč chtějí žáci studovat jazyky. Dále rozdáme studentům dotazníky, aby zaškrtli co nejpřesnější odpověď. Takový dotazník může vypadat podobně jako následující ukázka:

Chci studovat cizí jazyk, protože:

1. Může mi to pomoci pro získání dobrého místa.
2. Pomůže mi to porozumět anglicky mluvícím lidem a jejich způsobu života.
3. Kdo umí cizí jazyk, může si získat respekt ostatních.
4. Umožní mi to seznámit se a mluvit se zajímavými lidmi.
5. Potřebuji to pro svou práci.
6. Umožní mi to myslet a chovat se jako anglicky mluvící.

Na základě tohoto dotazníku sestaví učitel plán, podle kterého bude vyučovat v konkrétní skupině.

2.1.8.2 Rádce, poradce

Jiná role v komunikativní výuce jazyků je role rádce, či poradce. Učitel je pro žáky vzorem efektivní komunikace, tak žáky podněcuje pomocí zpětné vazby.

S první úlohou, kterou dá učitel svým žákům k vypracování, začíná proces, vedoucí k vlastním a nezávislým znalostem a schopnostem v užívání cizího jazyka. Vyučující si musí být od samého počátku vědom toho, že zadané úlohy (ať jsou určeny k individuálnímu zpracování nebo k práci ve skupině) jsou zárodkem vývoje jazykové nezávislosti žáka. Po skončení školní docházky

by měla být schopnost samostatné práce již pevným zvykem. Od této chvíle odpovídá žák za svou práci v plné míře sám, učitele už nepotřebuje. Úspěšné ukončení tohoto procesu však vyžaduje cílevědomou pedagogickou práci, vychovávající k samostatnosti a zodpovědnosti.

2.1.8.3 Manažer skupinových procesů

Ať vyučující postupuje ekonomicky a se svým časem zachází starostlivě nebo ne, stejně se mu nepovede jednotlivým žákům početné třídy věnovat po delší dobu osobní pozornost. S tímto problémem se lze vypořádat, když už třídu rozdělíme do několika pracovních skupin. Probranou látku tak můžeme intenzivněji procvičovat. Práce ve skupinách ovšem znamená ulehčení práce pro jednotlivého žáka. Jestliže má získat více než povrchní znalosti jazyka, je i nadále potřeba usilovně cvičit a pozorně se věnovat opakování. Vnitřní monolog myšlenek musí stavět na praxi externího dialogu diskuse s ostatními. Po práci ve skupinách by vždy měla následovat fáze reflexe, aby se výsledky kolektivní práce mohly „vstřebat". Se vzrůstajícími zkušenostmi a jazykovými schopnostmi se nutná ústní cvičení přesouvají na pole práce dvojic nebo skupin, podle potřeby pod nepřímým dozorem učitele nebo zcela mimo jeho možnost ovlivňovat hovor.

V komunikativní výuce jazyků přestává být učitel tím, kdo řídí všechny procesy probíhající při výuce. Jeho rolí je organizovat žáky a komunikaci mezi nimi. Na to si však těžko zvykne učitel, který jako svou základní úlohu vidí ve vyhledávání a opravování chyb a zadávání testů.

2.1.9 Role výukových materiálů

Pro komunikativní výuku jazyků lze použít celou řadu materiálů. Jejich hlavní úlohou je zprostředkovávat živý jazyk užívaný při komunikaci. Zaměříme se na tři druhy materiálů: text, úkoly a reálie.

- *Textové materiály*
  Existuje mnoho učebnic, které spojují a podporují komunikativní výuku. Jejich osnovy často nejsou podobné standardním osnovám a strukturovaným organizovaným osnovám. Některé byly napsány v rámci většího celku, sylabu s malou dávkou komunikativního vyučování jazyků. Některé učebnice například nemají žádné obvyklé dialogy, drilování a namísto toho používají kazetové nahrávky, výuku spojenou s obrázky a konverzaci. Skupinová práce se skládá ze dvou odlišných textů pro skupinovou práci,

každá obsahuje odlišnou informaci potřebnou pro skupinovou aktivitu každého z páru.
Cizímu jazyku se učíme z různých důvodů a k dosažení různých cílů. Samozřejmě, silně motivovaný dospělý student, který studuje jazyk jako takový, může opakovaně číst určité texty a hodně se o jazyku naučit. Taková osobnost, zabývající se jenom jazykem samotným, může ignorovat jeho obsah. Děti jsou oproti tomu málokdy schopné pozorovat obsah a formu odděleně, jejich zájem o obsah textu splývá s jeho jazykovou formou. Učebnice je tedy čistě funkcionálním prostředníkem mezi dítětem, jeho představivostí a zájmem na jedné straně a mezi zdánlivě nedosažitelnými záhadami ukrytými za bariérou cizího jazyka na straně druhé.

- *Materiály se zaměřením na úkoly*
  Variace různých her, hraní rolí, simulace a úkolové komunikativní aktivity jsou také součástí vyučovacích hodin komunikativní výuky. Typicky to jsou tyto: cvičební sešity, párové komunikativní materiály, studentské interaktivní cvičební knihy. Často jsou informace kompletní až když partneři dokončí obě části, jinak postrádají smysl.

- *Reálie*
  Mnoho zastánců komunikativní výuky jazyků upřednostňuje, jsou–li při výuce využívány materiály autentické, „ze života". Jde o noviny, časopisy, inzeráty, mapy, grafy atd. Další objekty mohou podporovat mluvní cvičení, např. plastický model k výkladu světových stran.

2.1.10 Způsob práce

Vyučujícímu se při zadávání úkolů nabízí příležitost — a to jak ve třídě, tak doma — naučit žáky metodické práci. Pozvolným navykáním žáka na práci bez dohledu a v roli přítele a spolupracovníka, nikoli nepřátelsky smýšlejícího kritika, pomáhá učitel jednotlivcům přebírat odpovědnost za jejich mimoškolní činnost. Žák se učí odpovědnosti za jejich plánování a za svědomité splnění úkolů, odpovídajících tak kritériím pracovního kolektivu a pochopí, že úkol není konečnou stanicí, ale začátkem jeho osobního pokroku a výchozím bodem stabilizace naučených jazykových vazeb. Vyučující by žákům za žádnou cenu neměl dovolit, aby se vrátili ke starým návykům a cítili odpovědnost jen vůči učiteli, to znamená odpovědnost negativní — jednoduše řečeno strach z potrestání. Jako učitelé se musíme postavit na stranu žáka, teprve potom vidíme problémy z jeho zorného úhlu. Za těchto okolností je můžeme řešit

společně, jako jeho spojenci a ne jako hlídači, kteří čekají na to, aby našli žákovi nějakou chybu.

Nejenom třída, ale i škola mohou srůst dohromady a být jednou sociální jednotkou. Učební plán je v tom případě sestaven s ohledem na pečlivou integraci všech oborů. Učitelé mohou kupříkladu přidělit jednotlivým žákům nebo skupinám jednoduché vědecké práce, opírající se o jiné prameny. Výsledky jsou pak předvedeny celému třídnímu kolektivu.

Protože komunikativní principy mohou být aplikovány na učení jakékoliv úrovně, můžeme popsat, jak postupovat při práci v komunikativní výuce jazyků. Jedná se o souhrn ze široké rozmanitosti třídních aktivit a různých cvičení popisovaných v literatuře (2; 3; 4; 14).

1. Prezentace krátkého dialogu nebo několika mini–dialogů motivovaných a diskutovaných na běžných lidských situacích, rolích, témat a neformálnosti či formálnosti jazyka podle žádané situace a funkce.

2. Ústní cvičení každého projevu, části dialogu jsou prezentovány týž den (veškerá opakování v hodině, dělení třídy na půlky, skupiny, individuální výuka), zpravidla předcházený vaším modelem. V krátkých rozhovorech jsou použita podobná cvičení.

3. Otázky a odpovědi založené na předmětu rozhovoru a samotné situaci.

4. Otázky a odpovědi mající vztah ke studentovu osobnímu zážitku, který je situován okolo tématu rozhovoru.

5. Studium jednoho základního komunikativního vyjádření v dialogu nebo jedné konstrukce, kterou osvětlíme na příkladu.

6. Student objevuje zobecnění základních pravidel. Může obsahovat toto: jeho ústní a psaná forma, jeho pozice v projevu, jeho formální či neformální úroveň, jeho gramatické vyjádření a význam.

7. Ústní zjišťování, vysvětlující aktivity.

8. Ústní předvedení aktivity — pojednání, řízení k volnějším komunikativním aktivitám.

9. Kopírování dialogů a útržkovitých rozhovorů pokud nejsou ve vyučovací hodině.

10. Vybírání vzorků domácích úkolů, pokud byly zadány.

11. Ohodnocení vědomostí (pouze slovně).

2.2 Metody využitelné při výuce skriptovacího jazyka PHP

Metoda je postup směřující nás záměrně k určitému (vytyčenému) cíli. Při výuce je cílem naučit se řešit samostatně úkoly v dané problematice. V programování je to zvládnutí programovacího jazyka. Metod může být několik, některé více, jiné méně efektivní. Dají se různě kombinovat. Každému žákovi vyhovuje něco jiného, každý má jiné individuální potřeby. V dalším textu se budu zabývat tím, jaké metody se dají využít se zřetelem na komunikativní výuku programovacího jazyka. Ty jsou převzaty z komunikativní výuky cizích jazyků a jiných pramenů (2; 7; 9; 13; 14) s důrazem na aplikovatelnost při výuce skriptovacího jazyka PHP. Bohužel neexistuje žádná dostupná typologie, proto uvedu pouze výčet metod, které se dají použít a jsou vhodné pro výuku skriptovacího jazyka PHP. Vhodné mi přišly následující metody, jelikož se dají aplikovat z komunikativní výuky jazyků na komunikativní výuku programovacího jazyka:

- dialogické metody,
- heuristická metoda,
- problémová metoda,
- fixační metoda,
- didaktické hry,
- ilustrační metoda,
- demonstrační metoda,
- metody samostatné práce,
- výkladová metoda.

V dalším textu se zaměřím na každou metodu zvlášť a vysvětlím, k čemu se která využívá nejčastěji.

2.2.1 Dialogické metody

Je to například diskuse, beseda nebo dialog. Předpokladem jsou minimálně dva účastníci, kteří jsou aktivizováni pomocí vhodně zvolených otázek nebo

formulací problému. Aby žáci mohli asociovat probírané učivo (budeme nazývat přípravnou fází) s dosavadními znalostmi (posuzují problém, navrhují řešení, znovu hodnotí řešení). Úlohou vyučujícího je navrhovat postupy, sledovat úspěšnost výuky a formulovat otázky podporující samostatné myšlení a rozvoj tvořivosti.

Diskuse může být velmi plodná, jestliže máme společnou základnu, tj. dostatek materiálu posbíraného v přípravné fázi. Dobře vedená diskuse objasní beze zbytku všechno, co předtím nebylo tolik srozumitelné. Je omyl věřit tomu, že když při prvním pročtení neporozumíme některým slovům (příkazům), a tak máme jenom všeobecný a mlhavý přehled o obsahu a myšlenkách textu, není třeba se snažit textu přesně porozumět. Každé nové pročtení s sebou přináší postupné porozumění stále většímu množství významových souvislostí. Každá přípravná fáze a každá diskuse po probrání učiva upevňuje získané znalosti v žákově vědomí.

Tyto metody jsou vhodné například pro úlohu hledání nejefektivnějšího algoritmu. Dále slouží k vyjasnění problematických oblastí učiva a často se používají k hodnocení práce v hodině. Součástí je formulace závěru, který shrnuje veškeré poznatky z diskuse a slouží ke korekci nesprávných a k zafixování správných řešení.

Dialogické metody jsou často kombinovány s výkladem, didaktickými hrami a demonstračními postupy. Jedním takovým příkladem může být osvětlení významu psaní přehledných PHP skriptů:

Každý žák napíše skript na předem stanovený problém, který určí každému žákovi učitel. Po vypracování si každý žák prohlédne výsledky prací svých spolužáků a snaží se odhalit, k čemu skript slouží. Své domněnky si každý zapíše a dále si je společně zkontrolují podle učitelovo zadání, zda-li správně odhadli funkci skriptu. Následuje diskuze o tom, jak se v kterém skriptu kdo orientoval a proč. Výsledkem je postupné zobecňování pravidel pro psaní přehledných zdrojových kódů samotných skriptů PHP.

Dalšími možnostmi je diskuse nad neúplným zdrojovým kódem (k čemu slouží) nebo skládání částí zdrojového kódu v jeden celek. Žáci mohou pracovat ve skupinách či samostatně, musí však následovat společná diskuse nad danou úlohou. Dialogická metoda je využita v kapitolách 3.3, 3.4 a 3.6.

2.2.2 Heuristická metoda

Vyžaduje hledání a především ověření nových poznatků samotnými žáky. Nejčastěji se jedná o změnu jednoduchého, již napsaného skriptu. Tato metoda je používána pro objevování nových obecných zákonitostí i konkrétních vlast-

ností jevů. Lze provádět tak, že pomocí promyšlených otázek je žák veden k cíli — k novým poznatkům. Otázky musí být vedeny logicky, následující otázka musí být spjata vždy s předešlou otázkou. Otázky také nesmí být nad síly žáků. Samotné provedení je časově náročnější než u dialogických metod (příprava PHP skriptů, náměty na změnu skriptů), ale poznatky jsou trvalejší díky zapojení složitějších myšlenkových pochodů a ověření v praxi (samotné spuštění skriptu). Pro tuto metodu se zvlášť hodí interaktivní prostředí osvětlené v kapitole 4. Je zde velmi důležité ukorigovat žáky pomocí připravených příkladů, aby slepě nezkoušeli nesmyslné příkazy.

Příkladem může být skript, který vygeneruje stránku s nápisem „Ahoj světe". Postupné přidávání funkcí na změnu výpisu a následné zobrazení výsledné HTML stránky je vhodné na první seznámení se skriptovacím jazykem. Nemusí se však zůstávat pouze u jednoduchých příkladů, pracovní aplikace (více v kapitole 4 na straně 62) je navržena tak, aby byla vhodná pro zkoušení jakýchkoliv skriptů. Heuristická metoda je použita v kapitole 3.2.

2.2.3 Problémová metoda

Je zaměřena na řešení složitějších úloh, které mohou přesahovat rámce jednotlivých hodin, ba dokonce i předmětu. Žáci řeší takový problém, jehož algoritmus řešení jim není znám. Děje se tak v těchto krocích:

1. krok — vymezení a pochopení problému,
2. krok — analýza problému (oživení všech poznatků, které souvisí s danou problematikou z vlastních sešitů, učebnic či jiné literatury),
3. krok — formulace hypotézy, myšlenkový experiment, předběžné řešení,
4. krok — ověření hypotézy (srovnáním s literaturou, s praxí, s učebnicí, u učitele, podle výpočtu...),
5. krok — shrnutí výsledků, stanovení závěru.

Problémová metoda může být použita na různé úrovni jako problémový rozhovor, samostatné či skupinové řešení nebo jako projektová metoda.

Problémový rozhovor se týká pouze diskuse nad danou úlohou, kdy není známo řešení. Jde o nejjednodušší variantu (z časového hlediska) problémové metody. Samostatné či skupinové řešení problému je ještě možné zvládnout za vyučovací jednotku. Projektová metoda je komplexní řešení, jejíž použití zabere delší časový prostor (více jak jedna vyučovací jednotka), záleží na zadání projektu.

K dosažení cíle v projektové metodě je nutné vyřešit několik dílčích úloh, ze kterých se celá úloha skládá. Tato metoda je zaměřena na rozvoj osobnosti žáka. Vzhledem ke svému rozsahu se jedná o náročnější práce jak pro žáky, tak pro učitele. Může se také jednat o společný projekt skupiny, kdy má každý vypracovat samostatně určitou část projektu. V tomto případě je nutné, aby někdo koordinoval celou skupinu (je určen vedoucí) a pomáhal v rozdělení činností, v důsledku odlišnosti jednotlivých členů skupiny (lepší a horší jedinci). Všichni jsou zodpovědní za konečný výsledek práce, proto je nutné vytvořit kooperaci a kladně motivovanou skupinu. Vyučující působí jako poradce, uvádí možné zdroje informací, pomáhá koordinovat činnost v rámci skupiny. V tomto typu úloh se rozvíjí samostatnost žáků, jejich tvořivost, vzájemná spolupráce a odpovědnost za vlastní jednání. Charakteristickým znakem je veliké přiblížení reálnému životu.

Příkladem může být vytvoření webového rozhraní pro čtení a psaní emailů z daného poštovního serveru. Skupina se rozdělí na vedoucího, programátory a grafika. Vedoucí skupiny dohlíží na celý projekt a skládá jednotlivé podúkoly v celek. Programátoři jsou rozděleni na tyto okruhy: posílání a čtení emailu, práce s poštovními složkami (vytváření, mazání, exportování), práce s jednotlivými emaily (přesouvání, kopírování, mazání, odpověď, přeposílání), správu adresáře vlastních kontaktů a grafika starajícího se o celkový vzhled stránek.

Výhody problémových metod jsou aktivita žáků, rozvoj samostatného myšlení a vedení k samostatnosti (žákovské projekty). Získané poznatky jsou trvalejšího rázu, žáci je dobře aplikují v dalším studiu. Nevýhodou je potřeba většího časového prostoru a velké náročnosti na didaktickou zdatnost učitele i na jeho přípravu. Příklady na problémovou metodu naleznete v kapitolách 3.4, 3.8 a 3.9.

2.2.4 Fixační metoda

Tato metoda se zaměřuje na opakování a upevňování zatím dosažených vědomostí a dovedností žáků. Může být vedena různými metodickými postupy, nejčastěji je to opakování se žáky pomocí promyšlených otázek (ptáme se na již probrané učivo) nebo to může být i písemné opakování (necháme znovu napsat podobný skript PHP, který byl vytvořen minulou hodinu). Účelem této metody není klasifikace žáků. Je vhodné tuto metodu zařazovat na závěr většího úseku učiva (tématického celku) předtím, než bude zařazena klasifikace z učiva. Učiteli může posloužit i jako významná zpětná vazba, kde si může ověřit, zda žáci učivo dokáží chápat v širších souvislostech, či potřebují ještě do-

datečné vysvětlení. Je účelné zařadit vyučovací hodinu, vytvořenou pomocí této metody, i v tématickém plánu předmětu programování.

Tato metoda není tolik náročná na přípravu, jelikož učitel již veškeré učivo, které chce opakovat, probral. Jediným dodatečným materiálem by bylo zařazení nových nebo podobných (předělaných) příkladů (skriptů) v PHP. Učitel musí dbát na opakování pomocí podobných příkladů, které si předem připraví. Může použít příklad nebo konstrukce podobné těm, co uváděl minulou hodinu. Tím si žáci zafixují jejich použití. V kapitolách 3.6 a 3.8 je možno vidět použití fixační metody.

2.2.5 Didaktické hry, inscenace a dramatizace

Význam těchto metod spočívá v možnosti přenášet poznatky zábavnou formou a intenzívně povzbuzovat zájem žáků o učení. Jsou využívány nejvíce pro navození uvolněné a zároveň kvalitní učební atmosféry na začátku nebo v průběhu hodiny, uvedení do nového tématu a opakování či procvičování učiva. Rozvíjejí komunikační dovednosti, kooperaci a pozitivní vztahy mezi žáky. Tyto metody jsou používány v kombinaci s jinými postupy (dialogickými) a jsou důležitou součástí komunikativního přístupu.

- *Dramatizace hry ve škole*
  Učitelé, kteří nikdy žádnou pečlivě připravenou a uvedenou dramatizaci ve škole neviděli, si těžko dovedou představit, jaké možnosti v sobě tento komunikativní prostředek skrývá a jaký blahodárný účinek má na žáky, které se hry zúčastní. Pravděpodobně by byli překvapeni, čeho všeho se dá se spontánně zpracovanými a hranými dramaty nebo žáky napsaným a realizovaným představením dosáhnout.

Příklady využití této metody je možno vidět v kapitole 3.8.

2.2.6 Ilustrační metoda

Význam této metody je často podceňován, přestože ji aplikují i tradiční přístupy vyučování. Vhodným použitím náčrtků, schémat a grafů nám pomáhá znázornit složité či abstraktní jevy, které jsou obtížně zachytitelné slovním popisem. Nesporná výhoda je v rychlosti a názornosti zobrazení.

Důvodem neúspěchu při učení se jazyku je obyčejně selhání představivosti. Za prvé je to selhání učitele, který si nedovede představit, že za určitých podmínek se žák nedovede vžít do situace, ve které ho učitel chce mít, a za druhé je to nedostatečná představivost žáka samotného. Není prostě schopen si na základě slov nebo obratů určité situace nebo obrazy vybavit. Většina učitelů vidí

východisko v upotřebení vizuálních pomůcek v té či oné formě a chtějí tak žáku zprostředkovat (imaginární) zkušenosti ze světa za stěnami učebny. Příliš málo učitelů realizuje možnost užívat těchto pomůcek s fantazií. To znamená, že předvádějí předměty a situace v ilustracích, ve filmu nebo jinak vizuálně zpracované, jako by to byly předměty a situace skutečné. Ale i nejlepší a sebe jasnější zobrazení zůstává zobrazením, naléhavě si žádajícím vztah a zapojení do rámce osobních zkušeností žáka. Toho se dá dosáhnout jenom za pomoci vědomého zaměstnání představivosti, ať už je její zatížení jakkoli minimální. Jak vyučující žákovi s vybavováním určitých představ pomůže? Poukazuje na paralely s jeho vlastními prožitky a dává mu příklady z okruhu jeho osobních zkušeností, až si je jistý, že žák pochopil.

Mapa asociací se používá při sběru informací o daném problému. Můžeme ji s výhodou zařadit při plánování projektů, kdy žáci sami navrhují, co vše bude pro daný projekt potřeba, co budou muset řešit a nakonec navrhnou celý postup řešení (rozdělení jednotlivých úkolů). Náčrtek kreslí postupně na tabuli, až vznikne přehled všech činností v projektu.

Příklady použití této metody jsou k dispozici v kapitole 3.9.

2.2.7 Demonstrační metoda

Principem demonstrační metody je především předvedení či požadavek, aby někdo něco udělal. Může to být i požadavek, aby někdo něco vysvětlil poté, co je to jemu samému předvedeno. Učitel je v prvním případě ten, kdo demonstruje novou látku (použití nových programovacích konstrukcí ve skriptovacím jazyce). Ve druhém případě je demonstrujícím žák, kdy sám žák vysvětluje (demonstruje) své nově zažité znalosti či zkušenosti. Zřídka se používá samostatně, většinou jako doplněk jiných postupů výuky. Používají je vyučující i žáci pro vytváření podnětů v problémových metodách, názorné ověřování výsledků nebo jako doprovod diskuse. Nejvýznamnější vlastnosti jsou názornost a čerpání poznatků z reálného života. Nevýhodou může být zahlcení žáků poznatky, při velkém množství materiálů, které vede k rozdělení pozornosti a nepochopení podstaty problému.

Demonstrační metoda se může zařadit například pro výuku řídících konstrukcí jako jsou příkazy *if*, *for*, *while*, *repeat* a jiné. Žáci se rozdělí na skupiny, přičemž každá skupina dostane za úkol se seznámit s novým příkazem. Učitel může každé skupině podle potřeby pomoci. Na závěr všichni demonstrují novou řídící konstrukci ostatním žákům společně s vhodnými příklady.

Další úlohou, která je využitelná ve všech programovacích jazycích, je výměna hodnot dvou proměnných. Může být s velkým úspěchem vysvětlena

na následujícím příkladu. Máme tři nádoby, nejlépe nějak odlišené (barevně či popsané), z nichž dvě jsou naplněny odlišnými tekutinami (popř. sypkým materiálem). Úkolem žáka je vzájemně vyměnit obsah dvou naplněných nádob. Po krátkém přemýšlení a fyzické manipulaci s nádobami žáci zjistí, že třetí nádobu musí použít, aby dosáhli požadovaného výsledku. Poté situaci převedeme na proměnné a žáci si uvědomí nutnost třetí proměnné při samotné výměně dvou proměnných. Je možno vysvětlit i slovně, avšak názorné (demonstrační) předvedení je lepší.

Demonstrační metoda je použita v kapitolách 3.1 a 3.7.

2.2.8 Autodidaktické metody a metody samostatné práce

Vyučující se stávají poradci a organizátory jednotlivých činností. Pomáhají nalézt zdroje poznatků a vytvářejí individuální učební plány. Výhodou je respektování odlišností jednotlivců, jejich stylu učení a tempa. Tyto metody rozvíjejí tvořivost, odpovědnost, schopnosti plánování a organizace vlastní práce, podle vymezeného času. Nevýhodou samostatné práce je nutnost naplánování činností, jinak hrozí nedokončení práce v termínu. Příkladem mohou být samostatné úkoly u problémových metod či samostatné práce žáků.

Kreativní samostatné práce by měli být vždy zařazeny do vyučovacího procesu při výuce PHP skriptu. Když je některý z žáků opravdu dobrý, svědomitý a umí pracovat bez pomoci učitele, můžeme mu dovolit vzít si takovou práci domů k dokončení. Nad takovými pracemi je vhodná společná diskuse a je také na místě kritika ostatních žáků.

Od samostatné práce se můžeme dostat ke společné práci, projektům, kde jsou žáci rozděleni do skupin a každá skupina má zadaný úkol. Vypracování se děje ve skupině, avšak každý žák pracuje na dílčím úkolu samostatně. Má jen možnost konzultace s ostatními ve skupině nebo s vyučujícím. Po ukončení práce představí vedoucí skupiny výsledky učiteli, popř. vyhotoví pro učitele jejich písemný seznam.

Použití této metody je v kapitolách 3.5 a 3.6.

2.2.9 Výkladová metoda

Nejedná se o komunikativní metodu, ale uvádím jí, jelikož se bez výkladu nové látky ve výuce programování neobejdeme. Učitel hovoří k žákům, vysvětluje a diskutuje. Žáci naslouchají, kladou případné otázky a reagují svými poznámkami. Je vhodné když učitel demonstruje vykládané na příkladech (demon-

stračnı́ metoda). Dochází tak k těsnějším vazbám v naučené látce při opětovném vybavování učiva.

Tuto metodu využijeme nejvíce při výkladu nové látky, někdy je tato metoda dokonce nezbytná. Nejčastěji se jedná o nejjednodušší a nejrychlejší přípravu učitele při výkladu nové látky s porovnáním s ostatními metodami. Tato metoda je zařazena do většiny kapitol (3.1, 3.2, 3.3, 3.4, 3.7).

2.3 Specifika výuky programování na střední škole

Vyučovat programovací jazyk jako takový nebo naučit žáky programování? Kdy začít učit programovat? To jsou otázky, kterými se budu dále zabývat.

2.3.1 Učit programování nebo programovací jazyk?

Jestliže se rozhodujeme učit programování, musíme si položit otázku, zda-li chceme naučit žáky daný programovací jazyk nebo chceme-li je naučit programovat. Oba přístupy se velmi liší.

Pokud chceme učit žáky programovacímu jazyku, neučíme je základy algoritmizace, ale jen samotné syntaxi daného jazyka. Máme to o mnoho jednodušší, jelikož již někdo před námi žáky programování naučil a my pouze ukazujeme možnosti daného programovacího jazyka (syntaxe, datové typy, řídící struktury, odlišnost od již známého programovacího jazyka...).

Pokud chceme naučit žáky programovat, musíme zvolit vhodný programovací jazyk. Pro tuto volbu existuje mnoho názorů, jaký programovací jazyk vybrat. Po pročtení literatury Rudolfa Pecinovského (11) a diskuzí k článkům autora, jsem dospěl k těmto zjištěním:

- programování je nutno učit v programovacím jazyce, který má maximálně jednoduchou a přirozenou syntaxi (neodpoutává pozornost žáků od vlastního řešení zadaného úkolu),
- při programování umožnit co nejdříve tvorbu programů.

Nejvhodnějším programovacím jazykem je zmiňován Baltík a Karel. Tyto programovací jazyky mají velmi důležité vlastnosti:

- žáci v něm mohou hned první hodinu tvořit (pro ně) netriviální programy,
- hned zpočátku, když ještě nic neumějí, mohou řešit úlohy, které jsou pro ně relativně zajímavé,

- příkazy v něm jsou česky.

Skriptovací jazyky (i PHP) nejsou vhodné pro učení se programování, jelikož syntaxe jazyka není jednoduchá (žáci si jí musí osvojit). Navíc příkazy jazyka jsou převzaty z angličtiny, proto je žák v jasné výhodě, pokud anglicky umí. Výhodnější by bylo, kdyby příkazy programovacího jazyka byly v přirozeném jazyce žáka (místo příkazu *if* by mohlo být použito *když*). Problémem „českého" programovacího jazyka je to, že pro něj obvykle není překladač či interpret.

2.3.2 Programování na základní škole

Mnoho lidí zastává názor, že programování by se měli zabývat pouze ti, kteří o danou problematiku mají evidentní zájem. Asi bude mnoho odpůrců, co řeknou, že všichni žáci nepotřebují umět programovat (natož již na základní škole). Dle mého názoru (a nejenom mého) by programování mělo patřit k základnímu vzdělání (11), jelikož rozvinuje schopnost samostatně uvažovat a schopnost samostatně řešit problémy.

Osobně bych na základní škole začal s jednoduchými programovacími jazyky, jako jsou již výše zmiňované Baltík nebo Karel, pro jednoduchost syntaxe. Po zvládnutí takového programovacího jazyka (naučení se základních programovacích principů) by již nic nebránilo přejít na jiný programovací jazyk (Pascal, skriptovací jazyk PHP a jiné).

2.3.3 Programování na střední škole

Pro programování na střední škole platí totéž co pro učení programování na základní skole. Pro větší vyspělost žáků se ale často začíná již s nějakým klasickým programovacím jazykem. To já považuji za nešťastné řešení pro větší složitost syntaxe takových jazyků. Nejlepším řešením se zdá naučit žáky základy algoritmizace v nějakém zjednodušeném programovacím jazyce a teprve poté začít vyučovat programování s využitím některého vyššího programovacího jazyka. Při výuce programování na vyšším stupni školy (vyšší odborná škola, vysoká škola) platí stejná pravidla.

Žáci by měli také možnost dané programy odlaďovat (pokud nastane chyba, upozorní nás, kde chyba nastala, případně o jakou chybu se jedná). Ve vývojovém prostředí většiny známých programovacích jazyků je k dispozici nějaký odlaďovací prostředek. V PHP máme také možnost ladit skripty. Pokud uděláme chybu, na výstupu uvidíme navíc i případné chybové hlášení, a to *kde* a *o jakou* chybu se jedná. Výpisy chybových hlášení jsou v angličtině a jdou různě potlačit či vynucovat si jejich vyobrazení v samotných skriptech.

2.4 Zvláštnosti při výuce programovacího jazyka

Pokud se učíme cizímu jazyku (např. angličtině), máme k dispozici velkou slovní zásobu a bohatou gramatiku. Máme možnost komunikovat v daném jazyce, hovořit na určitá témata a dále se zdokonalovat v řeči. Ve výuce programovacího jazyka je tomu jinak. Veškerá komunikace je prostřednictvím přirozeného jazyka, programovací jazyk stojí na pozadí a neslouží přímo k vyjadřování myšlenek. Ty jsou dány až celkovým zápisem programu a jeho činností.

2.4.1 Komunikace mezi učitelem a žákem

V oblasti komunikace ve škole dominují tyto základní komunikace: mezi učitelem a třídou a mezi učitelem a žákem. Vždy se jedná o přirozený jazyk (verbální komunikace) obou zúčastněných stran. Může se také jednat o neverbální komunikaci, kdy pomocí jednotlivých posunků a gest je sdělována určitá informace.

První typ (učitel — třída) bývá monologem učitele a má obvykle formu výkladu, vysvětlení, sdělení, návodu, pokynu, líčení, popisu, vyprávění, výzvy, napomínání či kritiky. Ve vztahu k nácviku schopnosti komunikace, jejímž základem je dialog a diskuse, jde o nevhodnou činnost. Nejčastěji jde o monolog, dialog nebo skupinovou komunikaci. Ve výuce programování se jedná o vysvětlování principů psaní skriptů či vysvětlování nové látky.

Druhý typ (učitel — žák) se zabývá vzájemnou komunikací obou subjektů. Při procvičování nebo zkoušení učiva jde nejčastěji o nácvik standadních programovacích technik nebo se dělají jednoduché skripty na dané téma. Dialog je učitelem řízený a jeho obsah je úzce omezen na procvičované učivo. Málokdy je dán žákovi prostor na formulaci a smysluplné obhajování vlastních myšlenek a názorů. V uvedeném typu by mělo být normální i individuálně komunikovat s žáky (učitel obchází jednotlivé počítačové stanice, komunikuje pomocí elektronické pošty). Tento druh komunikace by měl vést žáky k samostatnému kladení otázek, hledání cest směřujících k možným odpovědím. Nejčastěji se jedná o dialog nebo monolog.

Učitel by měl nabádat k další, neuvedené komunikaci mezi samotnými žáky. Ti by se měli soustavně rozvíjet ve vzájemné komunikaci mezi jednotlivci nebo mezi skupinami formou cíleného dialogu nebo diskuse.

2.4.2 Komunikace mezi žákem a počítačem

Zabýváme-li se čistou komunikací mezi žákem a počítačem bez přímé účasti učitele, máme na mysli komunikaci mezi počítačovým programem a samotným žákem. Jedná se o veškeré aplikace, se kterými můžeme přicházet do styku s počítačem (programy pro tvorbu a úpravu textu, výukové programy, programovací prostředky a jiné). Verbální komunikace je zde výrazně omezena.

Programovací jazyk má oproti přirozenému jazyku určité odlišnosti. Jednou z nich je množství lexikálních a syntaktických konstrukcí a z něj vyplývající rozsah vyjadřovacích prostředků jazyka. Programovací jazyk byl vytvořen za specializovaným účelem, a proto má menší rozsah výrazových možností. Další odlišností je to, že programovací jazyk používáme převážně v psané podobě.

Charakteristickou vlastností počítače jako účastníka komunikace jsou jeho předvídatelné reakce. Počítač reaguje na vnější podněty, přičemž jsou již předem stanoveny (naprogramovány) člověkem. Stroj tedy komunikaci neiniciuje. Jsou stanoveny mechanismy zpětné vazby podporující komunikaci těchto dvou subjektů (např. potvrzení správnosti dat při práci s formuláři, chybová hlášení a jiné).

Dalším rozdílem je využívání jazyka (programovacího a přirozeného) při samotné výuce. Při výuce programovacího jazyka nemůžeme mluvit samotným programovacím jazykem. Vyučujeme pomocí přirozeného jazyka a teprve sestavování skriptů vede k používání programovacího jazyka.

2.5 Hodnocení

Nedílnou součástí každé výuky je hodnocení. Zabývá se dvěmi základními otázkami: *co hodnotit a jakým způsobem*. V tradičním přístupu jsou posuzovány konečné výsledky vzhledem k nějaké normě. Není zde však zohledněn proces vzniku výsledků, ani jejich kvalita v porovnání s individuální úrovní žáka. Komunikativní přístup využívá komplexní hodnocení různých oblastí výuky. Celkové hodnocení můžeme složit z několika částí. První kritérium bude například funkčnost programu a zda řeší daný problém. Dalšími kritérii mohou být: komunikace programu s uživatelem (uživatelská jednoduchost, vřelost), efektivita vlastního algoritmu (rychlost zpracování) a zápis zdrojového textu. Při hodnocení tedy neposuzujeme pouze znalost syntaxe a sémantiky jazyka, ale klademe důraz na individuální schopnosti žáků. Před provedením hodnocení musíme stanovit, jaká část znalostí nebo dovedností bude posuzována a sdělíme její význam v celkovém hodnocení pro dané učební období.

Je také důležité jestli se bude hodnotit tradiční stupnicí (stupnice 1 až 5) nebo pomocí alternativního hodnocení (slovní, procentuelní). V komunikativním přístupu se obvykle prosazuje společné slovní a procentuální hodnocení v rámci individuální i sociální vztahové normy. Procentuální vyjádření zvyšuje přehlednost, je pro vyučující podstatně méně náročné na implementaci, umožňuje získávání statistických údajů v rámci učební skupiny nebo školy a také slouží pro zachování kompatibility se státním systémem. Slovní hodnocení (ústní či písemné) velmi dobře vystihuje osobité kvality žáků, pomáhá při odstraňování chyb (jeho součástí je rovněž identifikace problémových oblastí) a napomáhá vytvoření bližší vazby mezi vyučujícími a žáky (13).

Největší podíl na celkovém hodnocení při komunikativním přístupu k výuce PHP skriptu mají písemné práce (vytváření aplikací, skriptů). Speciálním případem hodnocení je posuzování výuky samotnými žáky. Učitel zjistí, co žáky zajímá a podle výsledků přizpůsobí výuku (zvolí vhodné žákovské projekty, které navrhují samotní žáci).

Kapitola 3

Návrh postupů při výuce PHP skriptu

Následující postupy ukazují využití metod nastíněných v části 2.2. Při samotné výuce PHP skriptu vycházíme z těchto předpokladů:

- žáci již umí pracovat se značkovacím jazykem HTML,
- žáci znají základy algoritmizace.

Než žáky začneme učit skriptovací jazyk PHP, měli bychom připomenout již známé záležitosti (předchozí zkušenosti v jiném programovacím jazyce, základy algoritmizace, HTML) a hlavně osvětlit nové. To se týká hlavně zpracování skriptu PHP na straně serveru. Tomu věnuji následující kapitolu 3.1. V dalších kapitolách se již dají použít zpracované sady příkladů, v kterých jsou používány komunikativní metody.

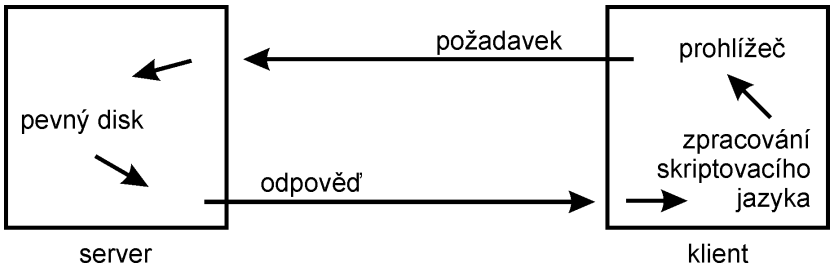
V následujících kapitolách 3.1 až 3.9 bude hojně využívána pracovní aplikace, která je popsána v kapitole 4.1.

3.1 Co je to PHP, jak pracuje

Rozsah výuky: 1 vyučovací hodina.
Požadované znalosti: žádné.
Znalosti po zvládnutí kapitoly: základní přehled o tom, co to PHP je, jak pracuje, a k čemu se používá.
Použité metody: výkladová a demonstrační metoda.

Skoro každý se již setkal s Internetem. Pomocí prohlížeče internetových stránek máme možnost prohlížet statické stránky, jako jsou různé archívy, ale také dynamické stránky. Ty mohou být nejrůznější povahy a jako příklad uvedu emailové servery, internetové obchody, zpravodajství a mnoho dalších.

Obrázek 3.1: Skript pracující na straně klienta



V dnešní době stále více webových serverů pracuje se skriptovacím jazykem PHP, který umožňuje vytvářet dynamické HTML (HyperText Markup Language) stránky (12), přistupovat k nejrůznějším databázím nebo dokonce vytvářet obrázky. PHP, což znamená „PHP: Hypertext Preprocessor" (1; 5), je široce používaný mnohoúčelový skriptovací jazyk, šířený pod Open Source licencí, zvlášť vhodný pro vývoj internetových aplikací a způsobilý pro vkládání do HTML. Velká část jeho syntaxe je převzata z programovacího jazyka C, Javy a Perlu. Cílem tohoto jazyka je umožnit webovým vývojářům rychle psát dynamicky generované stránky.

PHP je skriptovací jazyk pracující na straně serveru. To jej velmi odlišuje od skriptovacích jazyků jako je Java Script[1], JScript[2] a VB Script[3], které pracují na straně klienta.

Zobrazování internetové stránky probíhá tímto způsobem. Klient (samotný prohlížeč) vyšle požadavek na zobrazení stránky. Server se postará o poslání požadované stránky (přístup na pevný disk, přečtení souboru a odeslání). Ke klientovi se tak dostanou požadovaná data.

Zpracování skriptu na straně klienta se dá popsat následujícím způsobem (obrázek 3.1). Náš prohlížeč vyšle požadavek na získání dat ze serveru, kde se daná stránka nachází. Server přijme požadavek a odešle soubor, který se u něho nachází na pevném disku, klientovi. Klient přijme odpověď a soubor se dále zpracovává již podle určitého obsahu. Pokud je zjištěn nějaký skript pracující na straně klienta (např. Java Script), klientský počítač ho zpracuje. Nejčastěji to jsou soubory mající příponu *htm* nebo *html*.

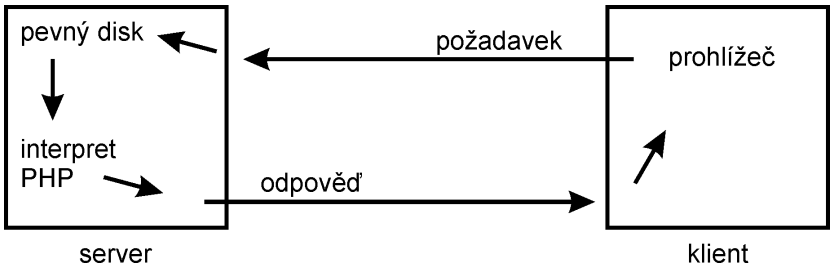
Oproti tomu zpracování na straně serveru popíšeme tímto způsobem (obrázek 3.2). Náš prohlížeč vyšle požadavek na získání dat ze serveru. Server přijme požadavek a zpracuje ho, což znamená, že přeloží skript, který vy-

[1]Skriptovací jazyk Java Script byl vyvinut společností Netscape Communications.

[2]Skriptovací jazyk JScript byl vyvinut společností Microsoft Corporation. Společně s Java Scriptem mají podporovat standart ECMA–62, což zajišťuje kompatibilitu v prohlížečích.

[3]Skriptovací jazyk VB Script byl vyvinut společností Microsoft Corporation. Podporují ho pouze prohlížeče Internet Explorer.

Obrázek 3.2: Skript pracující na straně serveru



generuje určitá data. Dále server vyšle odpověď (html stránku, obrázek) klientovi a ten jí zpracuje, přičemž v odpovědi může být také, jako v předchozím případě, skriptovací jazyk pracující na straně klienta. Nejčastěji to jsou soubory mající příponou *php*.

Zpracování jazyka PHP se tedy provádí přímo na serveru a ten, kdo si prohlíží stránky, nemůže vidět jejich zdrojový kód (rozumíme vše, co je uzavřeno mezi značkami označující začátek a konec skriptu). To co již na klientském počítači vidíme, je vygenerovaná stránka, kterou nám server (interpret jazyka PHP) vytvořil. Výklad demonstrujeme na příkladu *01_02.php* v pracovní aplikaci.

Na klientském počítači (lokálním) si necháme zobrazit skript. Ten vypíše aktuální čas ze serveru. Nyní se přesvědčíme, jestli je to čas přímo ze serveru. Změníme čas na lokálním počítači tak, aby ukazoval nesprávný čas. Poté znovu necháme zobrazit čas pomocí skriptu PHP. Žáci uvidí, že se stále bere aktuální čas a čas na lokálním počítači nehraje žádnou roli v zobrazování času ze skriptu.

A k čemu se PHP používá? Nejčastěji pro vytváření dynamického obsahu internetových stránek, dále pro pracování s daty z různých databází (nejčastěji MySQL), výpočty, počitadla přístupů, zpracování formulářů a jiných činností. Z českých serverů tento programovací jazyk používá například poštovní systém <http://email.cz>. Základní předpoklad pro psaní PHP skriptů je znalost HTML. Znalost jakéhokoliv skriptovacího jazyka pracující na straně klienta (Java Script, JScript či VB Script) není vůbec na škodu. Můžeme je totiž snadno využít pro kontrolu formulářů, než se samotná data pošlou na server a zakážeme odesílání prázdných formulářů na server.

3.2 První sada příkladů

Rozsah výuky: 4 vyučovací hodiny.
Požadované znalosti: žáci již umí pracovat se značkovacím jazykem HTML, znají základy algoritmizace.
Znalosti po zvládnutí kapitoly: základní přehled o vkládání skriptu PHP do HTML, možnosti výstupu PHP skriptu, seznámení s proměnnými a datovými typy, základní matematické operace, potlačení chybových hlášení.
Použité metody: heuristická a výkladová metoda.

Jelikož se heuristická metoda výrazně hodí pro zpracování prvních úkolů (viz. 2.2.2), pomocí kterých se žáci seznámí se základní syntaxí jazyka, využijeme ji pro seznámení se skriptovacím jazykem PHP. Budeme vycházet z prvního skriptu a dále budeme postupně přidávat další funkce. Ve výsledné části si zhodnotíme, co jsme se naučili.

Skriptovací jazyk PHP se nejčastěji vkládá přímo do HTML kódu. Žáci si otevřou internetovou stánku s pracovní aplikací (umístění na internetu nebo intranetu určí vyučující) a otevřou si příklad *01_01.php*.

Příklad *01_01.php*

```
<HTML><HEAD><TITLE>První</TITLE></HEAD><BODY>
<?php
echo "Ahoj světe!";
?>
</BODY></HTML>
```

První a poslední řádek jsou značky jazyka HTML, kterými se zabývat nebudeme[4]. Druhý řádek uvozuji značkou, která označuje začátek skriptu PHP (vkládání PHP skriptu přímo do HTML stránky). Třetí řádek je vlastní skript a poslední řádek ukončuje PHP skript. Příkaz *echo* vypíše na obrazovku to, co je uvozeno uvozovkami. Prohlížeč takovou stránku zpracuje a my nakonec uvidíme pouze text „Ahoj světe!" (pravý rám pracovní aplikace), což je v souladu s HTML standardem[5].

Aby si žáci uvědomili, že skript je zpracován opravdu přímo na serveru, vyzkoušíme si s nimi příklad *01_02.php*.

Příklad *01_02.php*

```
<HTML><HEAD><TITLE>První</TITLE></HEAD><BODY>
<?php
echo "Ahoj světe! <BR>";
```

[4]Pokud potřebujete pomoc, podívejte se do manuálu HTML (12).
[5]Ukázka si můžete prohlédnout na obrázku 4.1.

```
echo date("H:i:s",time());
?>
</BODY></HTML>
```

Tento příklad zobrazuje na prvním řádku text „Ahoj světe!" z předchozího příkladu a na druhém řádku informaci o aktuálním čase. Připomenu jenom, že se jedná o čas na serveru, proto nemusí být zcela korektní podle času na počítači, na kterém právě pracujete. Co jsme udělali s původním skriptem? Přidali jsme pouze následující řádek, který vypíše čas ve tvaru HOD:MIN:SEK oddělený dvojtečkami.

```
echo date("H:i:s",time());
```

Funkce *time()* vrací aktuální čas, bohužel v unixovém formátu (ve výsledné stránce by se zobrazilo jen nic neříkající číslo). Proto je zde použita ještě další funkce *date(řetězec,čas)*, která vrací formátovaný čas (či datum) podle daného řetězce[6] z funkce *time()*. Tento řádek vrací aktuální čas na serveru[7], přičemž příkaz *echo()* nám ho vypíše do výsledné stránky. Můžeme si všimnout, že v tomto příkladu je na řádku, který vypisuje pozdrav, přidána nepárová značka <BR>. Tato značka zalamuje řádek (viz. (12)). Pokud bychom první řádek neupravili, čas by se vypsal hned za dříve vypsaný text.

Žákům řekneme, aby si zobrazili zdrojový kód výsledné stránky (pouze pravé části rámce) a zjistili, co se ve výsledné stránce vypisuje. Uvidí, že je tam přímo zapsaná informace o čase. Pokud obnovíme stránku (tlačítko „Zobraz výsledek"), tak se skript znovu zpracuje a čas se zaktualizuje. Žáci uvidí tento (nebo podobný) výsledek (viz. obrázek 3.3).

HTML výstup z příkladu *01_02.php*

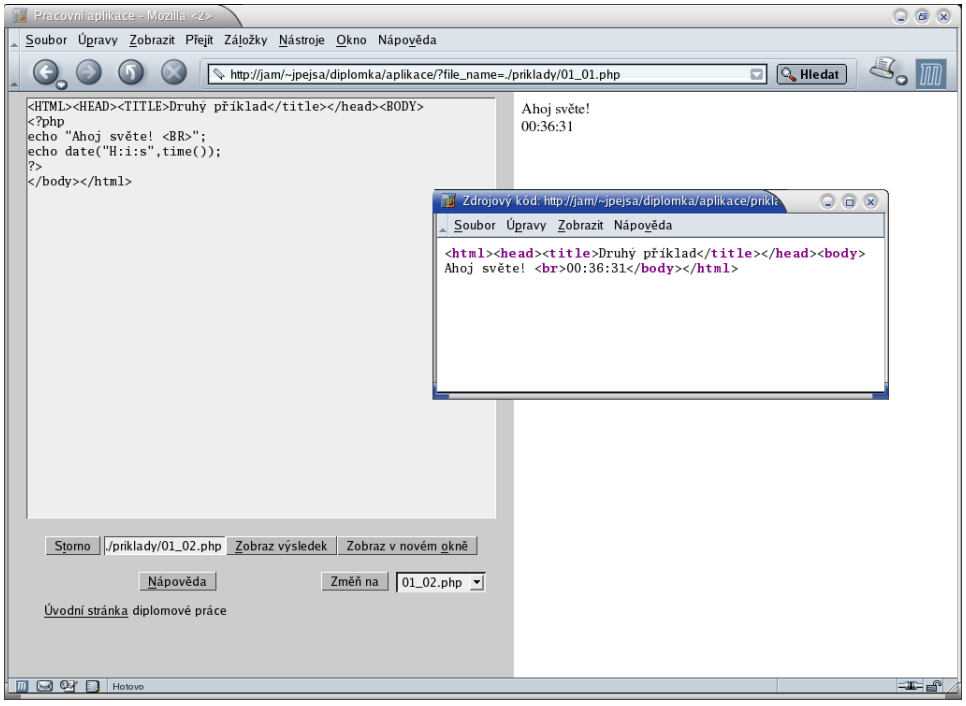
```
<HTML><HEAD><TITLE>První příklad</TITLE></HEAD><BODY>
Ahoj světe! <BR>16:24:06</BODY><HTML>
```

Pokud chceme zalamovat řádky ve zdrojovém kódu stránky, musíme nechat vypsat escape sekvenci `\n` v příkazu *echo* pro zapsání nového řádku. Tento nový řádek samozřejmě nebude mít vliv na výsledné zobrazení v prohlížeči, ale pouze na zdrojový text vygenerované stránky. Necháme žáky, aby si sami vyzkoušeli pomocí pracovní aplikace s použitím souboru určeného pro ukládání, *0.php*.

[6] Podívejte se do manuálu PHP (1) na možnosti formátování data popř. času a vyzkoušejte si různé výstupy.

[7] Že se jedná o čas vrácený serverem se můžeme přesvědčit tím, když si necháme zobrazit aktuální čas na našem lokálním počítači a porovnáme ho. Čas na našem počítači můžeme změnit a znovu porovnat s výstupem skriptu.

Obrázek 3.3: Ukázka příkladu 02_02.php



V dalším příkladu se naučíme pracovat s proměnnými (přiřazování, vypisování na obrazovku a sčítání numerických proměnných).

Příklad *01_03.php*

```
<HTML><HEAD><TITLE>První</TITLE></HEAD><BODY>
<?php
$str = "269 01, Rakovník";
$x = 3;
$a = 5.1;
$b = $x +  $a;
$y = $x + $str;
echo $b."<BR>";
echo "str: ".$str."<BR>";    // *1
echo "x: $x<BR>";            // *2
echo "y: ".$y;
?>
</BODY></HTML>
```

Třetí řádek přiřazuje proměnné `$str` určitý řetězec. V PHP se automaticky stává řetězcovou proměnnou. Ve čtvrtém řádku přiřazujeme proměnné `$x` číslo, podobně v dalším řádku přiřazujeme proměnné `$a` desetinné číslo (místo desetinné čárky používáme desetinnou tečku). PHP udělá typ čísla, v případě proměnné `$x` nastaví celočíselný typ, u proměnné `$a` nastaví reálný typ. V šestém řádku sečteme obě číselné proměnné a součet přiřadíme proměnné

`$b`. V dalším řádku sečteme dva různé datové typy[8]. Jediná podmínka je, že u řetězce musí být na prvním místě nějaké číslo, vše za číslem se ignoruje. Další čtyři řádky pouze vypisují obsah proměnných.

Máme samozřejmě možnost typy proměnných také přetypovávat. V příkladu *01_04.php* si ukážeme jak.

Příklad *01_04.php*

```
<HTML><HEAD><TITLE>První</TITLE></HEAD><BODY>
<?php
$a = 2.1728;
echo "typ proměnné a: ".gettype($a)."<BR>";
echo "výpis proměnné a: ".$a."<BR>";
if (settype($a, "integer")) echo "Přetypování úspěsné.";
else echo "Nastala chyba";
echo "<BR>";
echo "typ proměnné a: ".gettype($a)."<BR>";
echo "výpis proměnné a: ".$a."<BR>";
?>
</BODY></HTML>
```

Ve třetím řádku nadeklarujeme proměnnou `$a`, se kterou budeme pracovat. Automaticky se zvolí reálný typ čísla, typ *double*, podle čísla, které je proměnné přiřazeno (`$a` = *2,1728*). V dalších řádcích vypíšeme typ proměnné a její hodnotu. V šestém řádku přetypujeme proměnnou a testujeme, zda–li proběhla daná operace úspěšně (podmíněný příkaz znají žáci z jiného programovacího jazyka, syntaxe je tu velice obdobná; pokud chcete podrobnější popis podívejte se na (5)). Nakonec vypíšeme znovu typ proměnné a její hodnotu. Můžeme si všimnout, že se již jedná o jiný typ, *integer* (celočíselný typ). Necháme žáky, aby si vyzkoušeli přetypovat řetězec (typ *string*) na typ *integer*. Ať vyzkouší řetězec bez čísel a poté i s číslem na začátku. Můžeme si všimnout, že PHP přetypuje v jakémkoliv případě úspěšně.

Ke spojování řetězců jsme již operátor tečku (`.`) používali. Teď si ještě ukážeme několik případů, jak vypisovat obsahy proměnných.

Příklad *01_05.php*

```
<HTML><HEAD><TITLE>První</TITLE></HEAD><BODY>
<?php
$str="Řetězec";
$STR="text";
$c=$str . " " . $STR;   // spojení řetězců + mezera mezi
```

[8] V jiných programovacích jazycích musíme striktně dodržovat typy proměnných, v PHP nemusíme.

```
echo $c;
echo "<BR>";
echo ($c);
echo "<BR>";
echo "$c";
echo "<BR>";
echo "$cecilka je jméno";
echo "<BR>";
echo "$c ecilka je jméno";
echo "<BR>";
echo "${c}ecilka";
?>
</BODY></HTML>
```

První část kódu (řádky 3 až 5) spojí dva řetězce. Všimněte si, že PHP rozlišuje malá a velká písmena v názvech proměnných. Další řádky se zabývají výpisem řetězce. První možnosti jsou vlastně jen alternativy, jak vypisovat proměnné příkazem *echo*. Od řádku 12 však začíná zajímavější pojetí výpisu proměnné přímo z řetězce. Nejdříve se pokoušíme o vypsání proměnné `$cecilka`, což nevypíše nic, jelikož taková proměnná nebyla definována. Vypíše se až další text „je jméno" včetně mezer[9]. Dále vložíme jedinou mezeru a celý výstup se změní na „Řetězec text ecilka je jméno". Proměnná `$c` je již známá a proto se vypíše její obsah následovaný ostatním textem v řetězci. Poslední výpis je striktně zadané vypsání proměnné ohraničené znaky `{}`. Text který je mezi složenými závorkami se bere jako název proměnné. Nemůže tedy nastat situace, že by PHP nevědělo, co je a co není proměnná. Žáky necháme zase experimentovat a tvořit vlastní skript pomocí příkladu *0.php*.

Poslední příklad heuristické metody je věnován chybovým hlášení PHP a jejich násilného odstranění. Nesmí nás ovšem překvapit, že chybová hlášení jsou v angličtině.

Příklad *01_06.php*

```
<HTML><HEAD><TITLE>První</TITLE></HEAD><BODY>
<?php
echo (5/0);        // zobrazí chybovou zprávu
echo "<BR>";
echo @(5/0);       // nevypíse nic
?>
</BODY></HTML>
```

Třetí řádek zobrazí toto chybové hlášení *„Warning: Division by zero in ./01_06-.php on line 3"*[10]. Je to tím, že dělíme nulou. Pokud bychom chtěli toto varování

[9] Pro přesné zobrazení výstupu si zobrazte zdrojový kód výsledné stránky.

[10] Všimněte si, že PHP nás informuje velmi podrobně o chybě. „Varování: Dělení nulou v souboru 01_06.php na řádce 3".

potlačit můžeme použít znak @, který potlačuje varování a chyby PHP interpreta. Pátý řádek je již upraven a proto nevypíše nic. Nechte opět žáky experimentovat s daným příkladem. Ať zkusí vytvořit skript, který testuje tuto podmínku:

```
a je 5, b je 2;
pokud podíl a/b je roven 3, tak vypiš text "ANO"
```

Výsledný skript může vypadat například takto:

```
$a = 5;
$b = 2;
if ($a/$b==3) echo "ANO";
```

Tento příklad nic nevypíše, jelikož podmínka splněna není. Navíc si všimněte, že při testování podmínky musíme použít rovnítka dvě (na rozdíl od jiných programovacích jazyků). Zkusíme změnit hodnotu proměnné `$a` na 6 a ihned zjistíme, zdali je podmínka splněna.

Doporučuji ponechávat ve skriptech značky HTML `<HTML><BODY>` a `</BODY></HTML>`. Při práci se skupinou doporučuji vytvořit pro každého zvlášť adresář, kde bude mít každý žák svojí pracovní aplikaci. Při společné práci s jedním souborem *0.php* by mohlo docházet ke zbytečným komplikacím, daným ukládáním do jediného souboru více uživateli.

V dalším příkladu *01_07.php* si studenti vyzkouší jednoduché posílání vstupů (hodnot) skriptu pomocí formuláře. Žákům dáme k dispozici již hotový skript. Necháme je, ať si vyzkouší funkčnost daného skriptu a poté jim vysvětlíme, co je zde nového.

Příklad *01_07.php*

```
<HTML><HEAD><TITLE>Kalkulačka</TITLE></HEAD><BODY>
<?php
if (isset($_POST["operace"]))
  {
  echo "Byl odeslán formulář<br>";
  echo $_POST["a"];
  echo " ".$_POST["operace"]." ";
  echo $_POST["b"];
  echo " = ";
  switch ($_POST["operace"])
    {
    case "+": echo ($_POST["a"]+$_POST["b"]); break;
    case "-": echo ($_POST["a"]-$_POST["b"]); break;
    case "*": echo ($_POST["a"]*$_POST["b"]); break;
    case "/": echo ($_POST["a"]/$_POST["b"]); break;
```

```
    }
  echo "<BR><P>";
  }
?>
<FORM action="<? echo $PHP_SELF; ?>" method="POST"
 name="formular">
<INPUT type="text" name="a" value=5>
<SELECT name="operace">
 <OPTION value="+" selected="true">sečti</OPTION>
 <OPTION value="-">odečti</OPTION>
 <OPTION value="*">vynásob</OPTION>
 <OPTION value="/">vyděl</OPTION>
</SELECT>
<INPUT type="text" name="b" value=2>
<BUTTON type="submit" name="potvrzeni">Spočti!</BUTTON>
</FORM>
</BODY></HTML>
```

Nejprve zopakujeme co to je formulář je a k čemu se může používat (viz. předchozí znalosti HTML). Poté osvětlíme funkci celého skriptu. Skript si zkontroluje, zda-li byl poslán formulář (3. řádek). Pokud funkce *isset()* vrátí pravdu (proměnná `$_POST["operace"]` je nastavena), pokračuje se ve zpracování toho, co je uvozeno značkami „{" a „}" (podobně jako v pascalu klíčová slova *begin* a *end*). V případě, že formulář poslán nebyl (proměnná `$_POST["operace"]` není nastavena), tak se zobrazí jen formulář. Žáky necháme samotné zjistit proč se používá proměnná `$_POST` a také je necháme zjistit, jakou jinou metodou můžeme ještě posílat data z formuláře (jsou dvě možnosti POST a GET[11]). Nechte žáky změnit metodu poslání formuláře z POST na GET a nechte je vysvětlit, proč skript nefunguje. Nechte je, aby skript opravili tak, aby byl funkční s metodou posílání formuláře GET. Až se jim to podaří (změní všechny proměnné `$POST["proměnná"]` ve skriptu na `$GET["proměnná"]`) osvětlíme, k čemu se používá konstrukce *switch($proměnná)*. Všimněte si, že ve formuláři (značka `<FORM>`) je použit, v parametru *action*, výpis proměnné `$PHP_SELF`. Jedná se o globální proměnnou, která je k dispozici v každém skriptu a obsahuje cestu k souboru, který je zobrazen prohlížečem. Máme tedy zajištěno, že se formulář odešle znovu tomuto skriptu. Dále necháme žáky přidat další funkci pro zobrazení výsledku celočíselného dělení a jeho zbytku (k vydělení použijeme funkci *floor()*, která nám ořízne desetinná místa a ke zjištění zbytku po celočíselném dělení použijeme operátor %). Musíme přidat kód jak do části samotného skriptu (konstrukce *switch*), tak do formuláře, do části výběru požadované funkce (značka `<SELECT>` a `<OPTION>`).

[11] Více naleznete v (1; 12).

```
case "%": echo floor($_POST["a"]/$_POST["b"]);
          echo " (";
          echo ($_POST["a"] % $_POST["b"]);
          echo ")"; break;
<OPTION value="%">vyděl (2)</OPTION>
```

Určitě jste si řekli po chvíli experimentování, že není praktické pokaždé měnit hodnoty ze standardních (značka <INPUT>, parametr *value*), které se zobrazují ve formuláři. Určitě by bylo praktické, kdyby hodnoty, které zadáme, zůstaly ve formuláři i po jeho odeslání. Můžeme to udělat jednoduchým vypsáním příslušných hodnot rovnou při generování (vypisování) formuláře. Příslušné řádky tedy změníme takto:

```
<INPUT type="text" name="a" value="<? echo $_POST["a"];?>">
<INPUT type="text" name="b" value="<? echo $_POST["b"];?>">
```

Další úpravou bude ponechání vybrané operace po odeslání formuláře, namísto standardní první (sečti). To můžete zpracovat například takto:

```
<OPTION value="+" <? if ($_POST["operace"]=="+")
echo "selected=true";?>>sečti</OPTION>
```

Zkuste dělit nějaké číslo nulou. Zobrazilo se chybové hlášení? Jak se ho zbavíme? Podívejte se na stránce 37, jak se odstraňuje chybové hlášení! Na konečné zpracování skriptu pro základní matematické operace se podívejte na příklad *01_08.php* na stránkách pracovní aplikace.

Tento poslední příklad (*01_08.php*) byl ovšem již náročnější, a proto bych ponechal na učiteli, zda–li by ho zařadil do některé z prvních hodin. V této kapitole jsme se seznámili se základním psaním skriptů, jak vkládat skript do kódu HTML a s proměnnými.

3.3 Druhá sada příkladů

Rozsah výuky: 1–2 vyučovací hodiny.
Požadované znalosti: vkládání PHP skriptu do HTML, možnosti výstupu PHP skriptu, práce s proměnnými a datovými typy, matematické operace, funkce pro práci se soubory fopen(), fgets().
Znalosti po zvládnutí kapitoly: pochopení dodržování pravidel psaní přehledných skriptů, vkládání komentářů do skriptů.
Použité metody: dialogická a výkladová metoda.

Ve druhé sadě příkladů předvedu dialogickou metodu. Jedná se jednoduché počítadlo přístupů na stránku. Počet přístupů se bude ukládat do souboru *02_*, který se nachází na serveru v tomtéž adresáři jako daný skript. Aby naše počítadlo dobře fungovalo, musí mít soubor nastavena práva pro zápis do něj.

Žákům sdělíme zadání a řekneme, jaké jsou možnosti psaní komentářů v PHP skriptu:

Vytvořte počítadlo přístupů pro internetovou stránku. Počítadlo se při každém přístupu na stránku zvedne o jedničku.

Příklady zápisu komentáře:

```
$a = 20; // toto je komentář
$b = /* toto je jiný komentář */ 30;
```

Je potřeba, aby žáci již znali alespoň většinu funkcí nutných pro vytvoření počítadla. Jedná se především o čtení a zápis do souboru. Pokud studenti již všechny potřebné funkce znají, můžeme je ponechat vytvořit počítadlo samotné. V opačném případě vysvětlíme předem. Až žáci daný úkol zpracují, tak je přesadíme a necháme je, aby se zkusili zorientovat ve skriptu, který napsal jejich spolužák. Nejmenší problémy budou mít žáci s těmi skripty, které budou napsány přehledně. Navíc určitě studenta potěší, když nalezne ve skriptu od spolužáka komentáře, které mu říkají, co které řádky skriptu dělají.

Příklad *02_01.php*

```
<HTML><HEAD><TITLE>Počítadlo přístupů</TITLE></HEAD><BODY>
Jste <?php
$a=fopen("./02_", "r");$b=fgets($a,4096);$b=$b+1;
fclose($a);$a=fopen("./02_", "w");echo $b;
fputs($a, $b);fclose($a);?>. návštěník na této stránce!
</BODY></HTML>
```

Takto může vypadat jeden z vypracovaných úkolů. Pokud bychom neznali příslušné funkce, jen stěží poznáme, že se jedná o počitadlo přístupů, a jak vlastně funguje, nicméně výsledný skript je plně funkční. Až do druhého řádku se jedná o HTML kód. Poté je uvozena značka „<?php”, která uvozuje začátek PHP kódu. Přiřadíme do proměnné $a identifikátor souboru (celé číslo, pomocí kterého se provádí volání dalších funkcí) pomocí funkce *fopen(string jméno_souboru,string režim)*. Nejprve nastavíme režim čtení pomocí řetězce „r”. Funkcí *fgets(int identifikátor_souboru,int délka)* přečteme znaky ze souboru. Číslo 4096 zadané jako druhý parametr znamená, že přečteme maximálně 4 kB ze souboru. To, co přečteme uložíme do proměnné $b. Tím máme uloženo číslo poslední návštěvy (pokud existuje). K poslední návštěvě přičteme jedničku. Otevřený

soubor zavřeme funkcí *fclose(int identifikátor_souboru)*. Daný soubor znovu otevřeme, tentokrát však pro zápis (řetězec „w", který navíc zajistí zkrácení souboru na nulovou délku, soubor je tedy po otevření pro zápis prázdný). Dále vypisujeme hodnotu počtu přístupů. Pomocí funkce *fputs(int identifikátor_souboru,string řetězec)* vložíme do souboru daná data. PHP v této chvíli automaticky převádí z typu integer na typ řetězec (při přičítání jedničky se převádí z typu řetězec na typ integer). Znovu uzavřeme soubor (kdybychom soubor neuzavřeli, tak to PHP po přeložení skriptu udělá za nás). Nakonec ukončíme PHP kód značkou ?> a dopíšeme HTML stránku.

Skript není složitý, ale podívejme se na jiné řešení (velmi podobné), jak by mohlo vypadat jiné zpracování této úlohy.

Příklad *02_02.php*

```
<HTML>
 <HEAD>
  <TITLE>Počítadlo přístupů</TITLE>
 </HEAD>
 <BODY>
 <?
 $ident = fopen ("./02_", "r"); // otevreme pro cteni
 $pocet=fgets($ident,4096); // nacte data ze souboru
 fclose($ident); // uzavreme soubor
 $pocet++; // zvetsime hodnotu $pocet o 1
 $ident = fopen("./02_", "w"); // otevreme pro zapis
 fputs($ident, $pocet); // zapiseme hodnotu do souboru
 fclose($ident);
 echo "Jste ".$pocet.". návštěník na této stránce!";
 ?>
 </BODY>
</HTML>
```

Potřebovali jste k tomuto skriptu tak podrobný popis, jako k minulému? Určitě ne, jelikož bylo vhodně použito komentářů a navíc zápis celého skriptu je velmi přehledný. Je také lépe strukturovaný, neboť nejdříve zpracujeme čtení ze souboru a uzavřeme ho, zvýšíme počet přístupů a teprve potom zpracováváme zápis nové hodnoty do souboru. Poté vypisujeme výsledek. Správný zápis skriptu oceníme zvláště pokud pracujeme na nějakém delším skriptu, zvláště pokud se k němu vracíme po delší době. Žáky bychom měli vést k přehlednému zápisu skriptů, jelikož jako učitelé budeme muset jejich skripty prohlížet a zjistit jejich funkčnost.

Ve skriptech použitých v diplomové práci se dopouštím vědomě nesprávného zápisu skriptů. Důvodem je zobrazování celých řádků v tištěné verzi.

V pracovní aplikaci by bylo možné ponechat skripty v přehlednějším zobrazení celých dlouhých řádků, ale z důvodu studování diplomové práce a pracovní aplikace současně, nechávám tyto skripty totožné.

3.4 Třetí sada příkladů

Rozsah výuky: 4 vyučovací hodiny.
Požadované znalosti: práce s proměnnými, práce se soubory, funkce pro práci s cookies, funkce header(), příkaz exit, globální proměnná $REMOTE_ADDR.
Znalosti po zvládnutí kapitoly: různé řešení počítadla stránek.
Použité metody: problémová, výkladová a dialogická metoda.

Předchozí příklad počitadla přístupů využijeme v dalším příkladu, ve kterém je využívána problémová metoda. Velkou nevýhodou zaznamenávání počtů přístupů na stránku je to, že při každém načtení stránky se zvětší počet o jedna (jak jsme se přesvědčili v minulé kapitole). Když ale jeden uživatel vstoupí na naší stránku třeba desetkrát za minutu, přičte se za něj deset přístupů, což není vhodné. Máme tu hned problém, který necháme žáky společně řešit pomocí diskuze (dialogická metoda – viz. 2.2.1 na straně 18). Jedna možnost je použít cookies[12]. Cookies jsou data uložená HTML stránkou u klienta na pevném disku. Stránka, která je vytvořila je může kdykoliv znovu přečíst a dále s nimi pracovat. Naše rozšíření vytvoří cookies a pokud dojde k dalšímu načtení stránky a budou zjištěny cookies, tak se již počet přístupů nezvýší. Nevýhodou je, že klient si může cookies vypnout. Vycházíme z příkladu *03_01.php* což je totožný příklad s *02_02.php* a necháme studenty, aby sami zvládly tuto úlohu přepsáním zmiňovaného příkladu podle tohoto zadání:

Přepište příklad 03_01.php tak, aby se přístupy nepřipočítávali, pokud již uživatel byl na dané stránce. Použijte cookies.

Žáci se mohou setkat s těmito problémy: neznají funkce pro práci s cookies, funkci *header()* či příkaz *exit* (musí předcházet výklad). Dalším problémem je práce s hlavičkami v PHP. Ty se odešlou pouze pokud ještě skript nebo stránka neposlala žádný obsah (nesmí se poslat ani mezera či prázdný řádek). Pokud bude skript chybně napsán, PHP nás o tom upozorní chybovým hlášením, že již byla poslána nějaká data (a na jakém řádku) a že nemohly být aplikovány hlavičkové požadavky.

[12] Více naleznete v (1; 5; 8)

Příklad *03_02.php*

```
<?
 if (!isset($HTTP_COOKIE_VARS["pocitadlo"])) // pokud neni cookie
  {
  $ident = fopen ("./03_", "r"); // otevreme soubor pro cteni
  $pocet=fgets($ident,4096); // nacte data z otevreneho souboru
  fclose($ident); // uzavreme soubor
  $pocet++; // zvetsime hodnotu $pocet o 1
  $ident = fopen("./03_", "w"); // otevreme soubor pro zapis
  fputs($ident, $pocet); // zapiseme novou hodnotu do souboru
  fclose($ident);
  setcookie("pocitadlo","jiz jsi pristoupil na stranku",
             time()+60*60*24*365); // nastavime cookie
  header("Location:$PHP_SELF"); // do hlavicky HTML vlozime
                               // informaci, ze chceme presmerovat
  exit; // ukoncime provadeni skriptu
  }
?>
<HTML>
 <HEAD>
  <TITLE>Počítadlo přístupů</TITLE>
 </HEAD>
 <BODY>
 <?
 $ident = fopen("./03_", "r"); // otevreme soubor pro cteni
 $pocet=fgets($ident,4096); // nacte data z otevreneho souboru
 fclose($ident);
 echo "Jste ".$pocet.". návštěník na této stránce!"; // vypis
 ?>
 </BODY>
</HTML>
```

Cookies jsou součástí HTTP hlavičky, proto se funkce *setcookie()* musí volat před odesláním výstupu do prohlížeče. Navíc jsou přístupné až při dalším načtení stránky. Skript je navržen tak, že při prvním načtení neexistuje daná cookie (testování existence proměnné `$pocitadlo` na druhém řádku) a provede se přečtení souboru s počtem přístupů, přičte se jednička a zapíše se do daného souboru. Dále se zapíše cookie pomocí funkce *setcookie(string jméno,string hodnota,int platnost)*, do hlavičky HTTP protokolu se vloží informace o změně umístění (funkce *header(string)*, kde jako řetězec vložíme „Location: nové umístění") a nuceně ukončíme vykonávání skriptu příkazem *exit*. Při novém načtení stránky již existuje cookie, takže se první část nevykoná. Ve zbytku skriptu je jen otevření souboru, přečtení hodnoty a zobrazení výsledku na stránce.

Zadáme žákům další zadání:

Přepište příklad 03_01.php tak, aby se přístupy nepřipočítávali, pokud již uživatel byl na dané stránce. Pro zapamatování přístupu uživatele použijte IP adresu klienta a uložte si jí do souboru (03__) na serveru.

V tomto příkladu je použita možnost ověřování přístupů na stránku pomocí sledování IP adresy klientského počítače. Informace o IP adrese posledního návštěvníka se ukládá do souboru *03__*, informace o celkovém počtu návštěv do jiného souboru *03_*. Kontrola je využitelná v případě, že je na danou stránku relativně malý počet návštěvníků za den (maximálně do stovky přístupů). Nevýhodou je to, když budou současně dva klienti prohlížet tuto stránku (opakovaně si jí zobrazovat, znovunačítat), tak se budou střídavě připočítávat přístupy. První klient přistoupí, druhý klient přistoupí, první klient znovunačte stránku, druhý klient také — a jsou čtyři přístupy navíc. V porovnání s cookies jde o značnou nevýhodu, ale zase nezávisí na tom, zda–li má klient zapnuté nebo vypnuté cookies. Porovnejte s předchozím příkladem a příkladem *03_01.php*.

Příklad *03_03.php*

```
<HTML>
 <HEAD>
  <TITLE>Počítadlo přístupů</TITLE>
 </HEAD>
 <BODY>
 <?
 // cteni poctu navstev
 $ident = fopen ("./03_", "r"); // otevreme soubor pro cteni
 $pocet=fgets($ident,4096); // nacte data z otevreneho souboru
 fclose($ident); // uzavreme soubor
 // cteni IP adresy posledniho navstevnika
 $ident = fopen ("./03__", "r"); // otevreme soubor pro cteni
 $ip_adresa=fgets($ident,4096); // nacte data ze souboru
 fclose($ident); // uzavreme soubor
 echo "Předchozí návštěvník měl IP adresu $ip_adresa<BR>";
 echo "Vaše IP adresa je $REMOTE_ADDR<BR>"; // vypis
 // kontrola IP adresy, zvetsime hodnotu $pocet o 1
 if ($REMOTE_ADDR!=$ip_adresa) $pocet++;
 // zapis poctu navstev
 $ident = fopen("./03_", "w"); // otevreme soubor pro zapis
 fputs($ident, $pocet); // zapiseme novou hodnotu do souboru
 fclose($ident);
 // zapis IP adresy posledniho navstevnika
 $ident = fopen("./03__", "w"); // otevreme soubor pro zapis
 fputs($ident, $REMOTE_ADDR); // zapiseme hodnotu do souboru
 fclose($ident);
 echo "Jste ".$pocet.". návštěník na této stránce!<BR>";
 ?>
 </BODY>
</HTML>
```

O uvedených možnostech nechte žáky diskutovat. Určitě nabídnou i jiné možnosti (uložení všech IP adres místo poslední jedné). Určitě se žákům zmiňte

o možnosti ukládání přístupů do databáze. Při ukládání do databáze oceníme možnost si ukládat časy přístupů a informace o klientovi (verze prohlížeče, operačního systému, IP adresa). Žáky nakonec necháme rozhodnout, jak by se dala internetová počítadla vylepšovat. Jedním z návrhů může být zakázání připočítávání přístupů určitým IP adresám (například fulltextové vyhledávače cookies nemají a přistupují na stránky skoro denně). Žáci se mohou také zamyslet nad tím, co se stane, když klient bude mít vypnuté cookies. V tom případě mohou značně navýšit počet přístupů. Zamezení tohoto druhu nechtěného zvýšení existuje, pokud si zapamatujeme na určitou dobu IP adresu klienta (počítače), který k nám přistupoval a během této doby nezvyšovali přístupy od této IP adresy (viz. předchozí příklad).

V příkladu *03_01.php* se počet přístupů zvyšuje bez omezení, v příkladu *03_02.php* se zvyšuje pouze pokud není na klientském počítači nastavena cookie, v příkladu *03_03.php* se počitadlo zvyšuje pouze pokud je přistupováno z různých IP adres. V pracovní aplikaci si zkuste obnovit stránku, jestli se počet přístupů zvýší (u obou příkladů). U druhého zkuste smazat cookie a pak znovu obnovit stránku. U třetího příkladu zkuste přistupovat na stránku z různých IP adres (ze dvou počítačů zároveň) a pozorujte co se děje.

3.5 Čtvrtá sada příkladů — ilustrace různých technik čtení

Rozsah výuky: 1 vyučovací hodina.
Požadované znalosti: práce s proměnnými, příkaz include, práce se soubory a adresáři, psaní vlastních funkcí, práce s polem.
Znalosti po zvládnutí kapitoly: vkládání skriptů do jiného skriptu, psaní vlastních funkcí.
Použité metody: metoda samostatné práce, dialogická metoda.

V zásadě rozlišujeme tři techniky čtení (13). Detailní čtení, scanning a skimming. První technika detailního čtení spočívá, že žáci si přečtou daný skript a mají podrobně popsat, co přesně jaká funkce (skript) dělá.

Žáci dostanou následující dva skripty k prostudování. Následuje jejich samostatné vysvětlení, co který skript dělá. Postupují podle zadání:

Přečtěte si následující skripty a zjistěte, k čemu slouží. Po prostudování vysvětlíte vyučujícímu jejich funkce.

Příklad *04_01.php*

```
<?
function d ()
{
$mesice = array ("ledna", "února", "března", "dubna", "května",
"června", "července", "srpna", "září", "října", "listopadu",
"prosince");
return Date ("j") . ". " . $mesice[Date ("n") - 1] . " " .
Date ("Y") . ", " . Date ("H:i:s");
}
?>
```

Příklad *04_02.php*

```
<?
function f ($n)
{
if ($n == 0):
return 1;
else:
return $n * f ($n - 1);
endif;
}
?>
```

Vysvětlení by mělo vypadat přibližně takto: „V prvním příkladu zavolání funkce *d()* vrátí řetězec, který vrací dnešní datum a aktuální čas (např. „1. ledna 2003, 08:05:35"). Ve druhém příkladu se po zavolání funkce *f(int číslo)*, vrátí hodnota faktoriálu daného čísla."

Aby si žáci vyzkoušeli, zda-li bylo jejich vysvětlení daných funkcí správné, necháme je vyzkoušet si dané příklady pomocí pracovní aplikace a testovacího souboru *0.php*, kde si nechají vložit skripty a zavolají funkci, kterou chtějí otestovat. Výstupy obou příkladů si můžeme prohlédnout v příkladu *04_03.php* na stránkách pracovní aplikace (obrázek 3.4).

Příklad *04_03.php*

```
<HTML><HEAD><TITLE>Čtvrtý</TITLE></HEAD><BODY>
<?
include "./04_01.php";
include "./04_02.php";
echo "Dnes je: ". d()."<br>\n";
echo "Faktoriál čísla 5 je: ". f(5);
?>
</BODY></HTML>
```

Obrázek 3.4: Výstup příkladu *04_03.php*

Použití scanningu se hodí pokud chceme po žácích, aby rychle zjistili, co daný skript dělá. Zadání pro žáky je:

Prozkoumejte zběžně následující skript a zjistěte, co dělá. Vysvětlete funkci skriptu vyučujícímu. Na prozkoumání máte 5 minut.

Příklad *04_04.php*

```
<?php
include "../../head.php";
$pocet=0;
function adresar($dirr)
{
$handle=opendir($dirr);
echo "<TABLE width=100 border=0 cellspacing=0 cellpadding=0>";
while (false!==($file = readdir($handle))) {
    if ($file != "." && $file != ".." && $file != "index.php") {
        if (is_dir ($file))
  {
  echo "<TR><TD colspan=2 align=center><FONT size=13><B>
        $file</B></FONT>\n</TD></TR>";
  adresar("./".$file);
  }
else
  {
  $pocet=$pocet+1;
  if (($pocet/2)==round($pocet/2))
   {
   echo "<TR><TD><IMG src=\"$dirr/$file\" width=160 hspace=0
         vspace=0 border=0></TD><TD><A href=\"$dirr/$file\">
         $dirr/$file</A>\n</TD></TR>";
   }
  else
   {
   echo "<TR><TD><A href=\"$dirr/$file\">$dirr/$file</A>\n
         </TD><TD><IMG src=\"$dirr/$file\" width=160 hspace=0
```

```
            vspace=0 border=0></TD></TR>";
     }
    }
      }
}
echo "</TABLE>";
closedir($handle);
}
?>
<CENTER><H1>moje malá sbírka fotek...</H1>
<H5>...často aktualizovaná!!!</H5>
<TABLE width="100%" cellspacing="10" border="0" cellpadding="10"
align="center"><TR><TD align="center">
<?
adresar("../images"); echo "<br>";
?>
</TD></TR></TABLE></CENTER></BODY></HTML>
```

Žáci by měli podat následující osvětlení, i když neznají všechny funkce a nezkoumali podrobně daný skript: „Skript do sebe vkládá soubor *head.php* (v příkladu se jedná o standardní hlavičku HTML stránky). Dále zde dominuje funkce *adresar($dirr)*, která vypisuje jména souborů a zobrazuje je jako obrázky (značka `<IMG>`). Pokud tato funkce narazí na další adresář, volá se rekurentně. Jedná se tedy o zobrazení náhledů a odkazů na obrázky (dle značky `<IMG>` a jejím konstantním parametru *width*), které se nacházejí v nějaké adresářové struktuře (zde adresář *./images*). Navíc se obrázky zobrazují střídavě vlevo nebo vpravo společně s jejich názvem souboru."

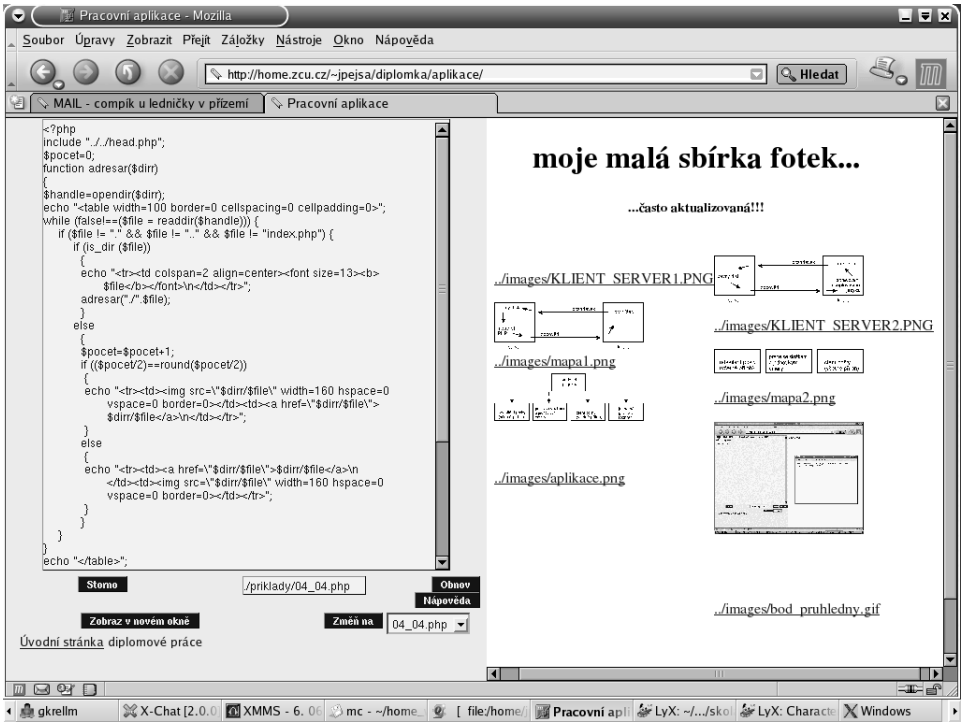
Vyzkoušejte si daný skript v pracovní aplikaci! Podívejte se na příklad výstupu na obrázku 3.5 v pracovní aplikaci.

3.6 Pátá sada příkladů — samostatné práce žáků

Rozsah výuky: 1–4 vyučovací hodiny (podle situace).
Požadované znalosti: práce s proměnnými, datové typy, funkce mail() pro odesílání pošty, HTML formuláře.
Znalosti po zvládnutí kapitoly: hlubší práce se soubory v PHP, samostatnost vyhledání potřebných informací.
Použité metody: metoda samostatné práce, dialogické metody, fixační metoda.

Při učení se nesmí zapomínat na takzvané „domácí úkoly", které musí žáci tvořit samostatně bez pomoci učitele. Učitel se stává poradcem a žáci mají možnost komunikovat sami se sebou nebo s vyučujícím, pokud jim něco brání v samostatné práci pokračovat. Učitel si musí připravit několik zadání (pokud bude dávat práci domů) nebo jen jeden příklad (pokud budou žáci pracovat

Obrázek 3.5: Příklad vyobrazení skriptu v prohlížeči



na úkolu ve škole při vyučovací hodině). Ve druhém případě je nutné zajistit opravdovou samostatnost samotných žáků. Následují zadání příkladů pro samostatnou práci žáků s vyřešením a vysvětlením skriptu.

Napište skript, který bude zajišťovat funkci „knihy návštěv" na dané stránce. K zadávání údajů (jméno, email, vlastní text zprávy) použijte formulář a odešlete znovu skriptu, který zprávu zpracuje a uloží do zvláštního souboru, kde se po každém přidání zprávy přidá kus zdrojového kódu HTML jazyka s obsahem již odeslaných formulářů. Zajistěte, aby poslední odeslaný formulář se zařadil na první místo ve výpisu. Výsledná stránka může vypadat např. jako na přiloženém obrázku[13].

Příklad *05_01.php*

```
<?
// zápis do souboru z formuláře
if (isset($_POST["jmeno"])) if ($_POST["jmeno"]!="")
 {
 $zapis="";
 $zapis.="<P align=justify>".date("j. n. Y (H:i)",time())."<BR>\n";
 $zapis.="Odesílatel: <A href=\"mailto:".$_POST["email"]."\">";
 $zapis.=$_POST["jmeno"]."</A><BR>\n";
 $zapis.="<I>".$_POST["text"]."</I></P>\n";
```

[13] Obrázek 3.6

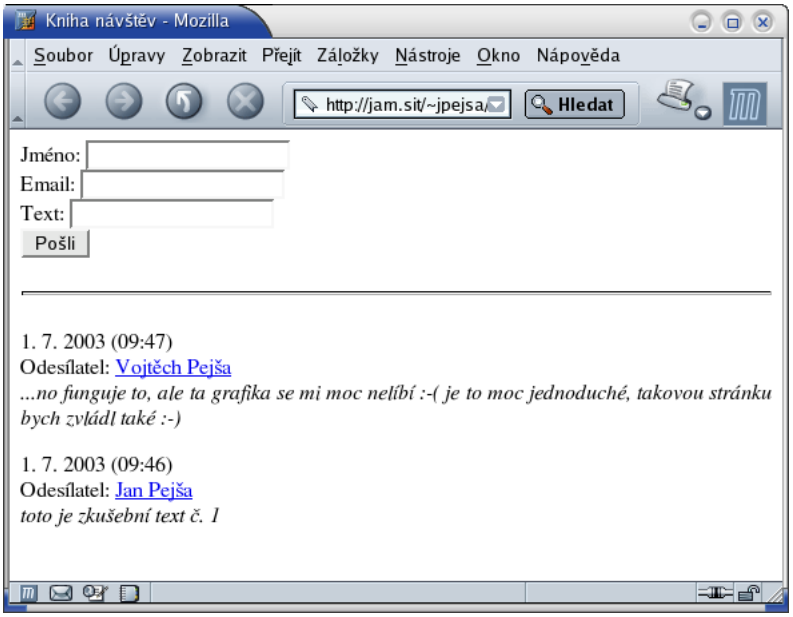
```
 $zapis.= join ("", file ("./navstevy"));
 $file = fopen ("./navstevy", "w");
 fputs($file, $zapis);
 fclose($file);
 header ("Location: $PHP_SELF"); exit;
 }
?>
<HTML><HEAD><TITLE>Kniha návštěv</TITLE></HEAD><BODY>
<FORM action="<? echo $PHP_SELF; ?>" method="POST">
Jméno: <INPUT type="text" name="jmeno"><BR>
Email: <INPUT type="text" name="email"><BR>
Text: <INPUT name="text" type="text"><BR>
<BUTTON type="submit">Pošli</BUTTON>
</FORM>
<HR>
<?php
// čtení ze souboru a vypsání na HTML stránku
readfile("./navstevy");
?>
</BODY></HTML>
```

Skript při načtení zobrazí formulář, pomocí kterého můžeme odesílat data znovu samotnému skriptu (řádek 17–24) a také vypisuje obsah souboru *navstevy* (řádek 27), který se nachází ve stejném adresáři jako skript. Řádky 1 až 16 zpracovávají odeslaný formulář. Nejdříve se testuje, zda–li byla poslána proměnná `$jmeno` (metodou POST), která je obsahem formuláře. Dále testujeme, jestli není proměnná prázdná (odeslání prázdného formuláře). Zbytek skriptu je práce s řetězcem (proměnná `$zapis`), který vytvoříme pomocí dat z odeslaného formuláře (proměnné `$_POST["jmeno"]` a jiné). Nakonec přidáme do proměnné `$zapis` celý obsah souboru *navstevy*. Funkce *file()* čte ze souboru a každý řádek uloží do pole, funkce *join()* spojuje prvky z pole do jediného řetězce, přičemž jako první parametr si můžeme zvolit, čím mají být prvky (řetězce) spojeny. My vkládáme prázdný znak. Nakonec otevřeme soubor pro zápis, zapíšeme celý řetězec do souboru a poté uzavřeme. Celou stránku znovu načteme pomocí hlavičky HTML, do které vložíme informaci o přesměrování na stejné umístění skriptu, aby se již upravený souboru zobrazil s novými daty. Jak může vypadat výsledná stránka si můžete prohlédnout na obrázku 3.6.

Můžeme udělat další zadání s malou obměnou (jednodušší verze) pro lepší zapamatování právě naučené látky.

Napište skript, který bude zajišťovat funkci „knihy návštěv" na dané stránce. K zadávání údajů (jméno, email, vlastní text zprávy) použijte formulář a odešlete znovu skriptu, který zprávu zpracuje a uloží do zvláštního souboru, kde se po každém přidání zprávy přidá kus zdrojového kódu HTML jazyka s obsa-

Obrázek 3.6: Ukázka výsledné stránky „Kniha návštěv"



hem již odeslaných formulářů. Zajistěte, aby poslední odeslaný formulář se zařadil na poslední místo ve výpisu. Výsledná stránka může vypadat např. jako na přiloženém obrázku.

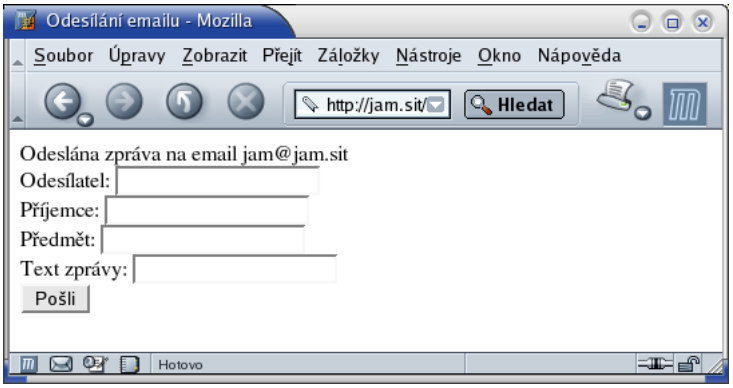
Zdrojový skript naleznete v pracovní aplikaci jako příklad *05_02.php*. V tomto příkladu je jedinou změnou způsob ukládání dat do souboru. Do proměnné `$zapis` se přiřadí pouze naformátovaná data z formuláře a rovnou se zapíší do souboru *navstevy*, který je v tomto případě otevřen pro zápis a ukazatel je nastaven na konec souboru (mód pro přidávání).

Napište skript, který bude odesílat data z formuláře (adresa odesílatele, adresa příjemce, předmět zprávy, text zprávy) na poštovní adresu příjemce. Využijte funkci mail().

Funkce *mail(komu,předmět,zpráva,hlavičky)* odesílá email zprostředkovaně pomocí serveru, kde je spuštěno samotný HTTP server s podporou PHP. Tento server musí mít ještě zajištěno odesílání emailů. Takovým serverem na ZČU je např. *home.zcu.cz*, kde mohou mít studenti ZČU umístěny vlastní internetové stránky. První parametr u funkce *mail()* říká, komu se bude email odesílat. Další dva parametry se vztahují k předmětu zprávy a k vlastní zprávě. Posledním (volitelným) parametrem jsou hlavičky emailu, kde můžeme specifikovat od koho je zpráva poslána, jakým klientem byla poslána a jiné volitelné parametry[14]. Funkce vrací hodnotu 1, pokud je provedena úspěšně (zpráva se podařila odeslat). Ve skriptu to také testuji a zobrazím případné chybové

[14]Více najdete v manuálu PHP(1).

Obrázek 3.7: Ukázka posílání pošty pomocí PHP



hlášení. Příklad úspěšně odeslaného emailu a grafického rozhraní internetového klienta pro posílání pošty vidíte na obrázku 3.7.

Příklad *05_03.php*

```
<?
if (isset($_POST["zprava"]))
 {
 if (mail($_POST["prijemce"],$_POST["predmet"],
          $_POST["zprava"],"From:".$_POST["odesilatel"]) == 1)
      echo "Odeslána zpráva na email ".$_POST["prijemce"];
 else echo "Neodesláno, nastala chyba při odesílání!";
 echo "<BR>";
 }
?>
<HTML><HEAD><TITLE>Odesílání emailu</TITLE></HEAD><BODY>
<FORM action="./05_03.php" method="POST">
Odesílatel: <INPUT type="text" name="odesilatel"><BR>
Příjemce: <INPUT type="text" name="prijemce"><BR>
Předmět: <INPUT type="text" name="predmet"><BR>
Text zprávy: <INPUT type="text" name="zprava"><BR>
<BUTTON type="submit">Pošli</BUTTON>
</FORM>
</BODY></HTML>
```

Obměna předchozího příkladu:

Napište skript, který bude odesílat data z formuláře na určenou adresu mobilního telefonu (telefonní číslo, zpráva). Využijte funkce mail(). Využijte možnost posílání zprávy na operátora, tvar emailu pro poslání je +420xxxyyyyyy@sms.eurotel.cz.

Zdrojový skript naleznete v pracovní aplikaci jako příklad *05_04.php*.

3.7 Šestá sada příkladů — databáze MySQL

Rozsah výuky: 4 vyučovací hodiny.
Požadované znalosti: podmíněný příkaz if, pojem databáze, syntaxe MySQL příkazů, HTML formuláře.
Znalosti po zvládnutí kapitoly: připojení k databázi MySQL pomocí PHP skriptu.
Použité metody: výkladová a demonstrační metoda.

V dnešní době patří databáze k neodmyslitelné části všech programovacích jazyků. Bez databází by jen těžko mohly vznikat internetové obchody, nejrůznější zpravodajské servery nebo celé firemní informační systémy. Proto i PHP umožňuje spolupracovat s databázemi. PHP má opravdu bohatou podporu databází. Jedná se o tyto: Oracle (verze 7 i 8), MySQL, mSQL, PostgreSQL a také Microsoft SQL server. Pomocí dalších modulů lze PHP dodat podporu pro libovolný typ databáze. V dalším textu proberu základní příkazy pro propojení PHP s MySQL.

Abychom mohli v PHP pracovat s databází MySQL, musíme se k ní připojit. V PHP existuje pro připojení k databázi funkce *MySQL_Connect(string počítač, string uživatel, string heslo)*. Jako počítač uvádíme adresu počítače, na kterém je puštěn server MySQL. V případě, že máte MySQL instalováno na lokálním počítači, je tato adresa *localhost*. Dále musíme mít přístup k databázi (tj. znát uživatelské jméno a heslo pro připojení k databázovému systému). Funkce vrací číslo spojení na databázi (podobně jako při práci se soubory v jiných programovacích jazycích).

Pro připojení k databázi se používá této konstrukce, která nám zjistí, zda připojení proběhlo v pořádku:

```
$spojeni = MySQL_Connect("localhost","uzivatel","heslo");
if(!$spojeni): echo "CHYBA: nelze navázat spojení!";
endif;
```

Pokud spojení proběhlo v pořádku, máme v proměnné `$spojeni` uloženo číslo spojení s MySQL, které budeme využívat v dalších funkcích. Vypíši základních přehled funkcí pro práci s databází MySQL[15] v PHP:

- *mysql_connect (string server, string uziv_jmeno, string heslo)*
  otevře připojení k databázovému serveru,
- *mysql_close (spojeni)*
  zavře otevřené připojení k serveru,

[15]Všechny použitelné funkce společně s podrobnějším popisem naleznete v manuálu PHP (1).

- *mysql_select_db (string jméno_databáze, spojení)*
  vybere jednu databázi, se kterou budeme pracovat,
- *mysql_query (string dotaz, spojení)*
  pošle dotaz MySQL, může se vrátit nějaký výsledek (dá se dále zpracovávat),
- *mysql_result (výsledek, spojení, typ výsledku)*
  načte výsledný řádek do pole,
- *mysql_fetch_array (výsledek, int typ)*
  načte výsledný řádek do asociativního nebo číselného pole.

K další práci bychom potřebovali znát příkazy (dotazy) specifické pro MySQL, které se používají při volání funkce *mysql_query()*. Popis těchto dotazů je k dispozici v manuálu MySQL (15). V každém skriptu se má ještě uzavřít připojení k serveru pomocí funkce *mysql_close()*. Pokud na to ve skriptu zapomeneme, PHP uzavře spojení za nás.

Žáci si otevřou příklad *06_01.php* a ihned uvidí co daný skript dělá. Podrobně žákům vysvětlíme, co který řádek skriptu dělá. Celý skript je složen z funkce *svatek(den,mesic)*, která má dva vstupy. Výstupem funkce je jméno toho, kdo má svátek v zadaném dni a měsíci, které určíme jako parametry. Funkce *svatek()* nejdříve vytvoří konekci k databázovému serveru (funkce *mysql_connect()*). Pokud konekce není vytvořena (z nějakého důvodu se to nepodaří), funkce vypíše chybové hlášení a ukončí se. Dále vybereme databázi, se kterou budeme pracovat (funkce *mysql_select_db()*). Přichází na řadu sestavit vhodný dotaz a poslat ho databázi. V proměnné `$sql` je takový dotaz[16] zapsán (chceme vybrat jméno, které se váže danému dni a měsíci) a v dalším řádku předáváme výsledek z funkce *mysql_query()* proměnné `$result`. Funkcí *mysql_fetch_array()* načteme výsledek do pole (proměnná `$vysledek` je datovým typem pole). Příkaz *return* zařídí, aby funkce vrátila dané jméno (z proměnné `$vysledek[0]`). Pokud nastane chyba je zařízeno, že se chyba detekuje funkcí *mysql_errno()*, která vrací číslo chyby nebo nulu, pokud vše proběhlo v pořádku. Zbytek skriptu je pouhé volání této funkce, kdy si nejdříve přiřadíme do proměnných dnešní den a měsíc v roce a poté necháme vypsat.

[16] Více o databázi naleznete v (15).

Příklad *06_01.php*

```
<?php
function svatek($den,$mesic)
{
// připojení k databázi
$Conn = mysql_connect ("localhost","jpejsa","rakovnik");
if (!$Conn) { echo "chyba při připojování k mysql"; exit; }
// výběr databáze
mysql_select_db ("user_jpejsa_db");
// pošleme dotaz
$sql = "SELECT Jmeno FROM Svatky WHERE Den=$den AND Mesic=$mesic";
$result = mysql_query($sql);
// Zobrazení výsledku v HTML
$vysledek = @mysql_fetch_array($result, MYSQL_NUM);
if (mysql_errno()==0) return $vysledek[0];
if (mysql_errno()!=0)
               return "<A title=\"Nastala chyba!\">???</A>";
// uzavřeme spojení
mysql_close ($Conn);
}
?>
<HTML><HEAD><TITLE>Svátek má...</TITLE></HEAD>
<BODY>
<?php
$den=date("j",time());
$mesic=date("n",time());
echo "Dnes je $den. $mesic. a svátek má ";
echo svatek($den,$mesic).".<BR>";
?>
</BODY></HTML>
```

Předchozí příklad byl zaměřen na výběr dat z databáze. Pokud ale nemáme žádnou tabulku v databázi vytvořenou, musíme se o to také postarat. Všechna data (příkazy jazyka SQL) jsou obsažena v souboru *06_*. Každý řádek představuje jeden příkaz, který se má provést. Příklad, jak vypadá tento soubor následující:

Příklad zápisu příkazů jazyka MySQL v souboru

```
CREATE TABLE Svatky (Den tinyint NOT NULL, Mesic tinyint NOT NULL, \
Jmeno varchar(50) NOT NULL)
INSERT INTO Svatky (Den, Mesic, Jmeno) VALUES (1, 1, '-Nový rok')
INSERT INTO Svatky (Den, Mesic, Jmeno) VALUES (2, 1, 'Karina')
INSERT INTO Svatky (Den, Mesic, Jmeno) VALUES (3, 1, 'Radmila')
..
INSERT INTO Svatky (Den, Mesic, Jmeno) VALUES (30, 12, 'David')
INSERT INTO Svatky (Den, Mesic, Jmeno) VALUES (31, 12, 'Silvestr')
```

První řádek (zde rozdělen na dva) vytvoří tabulku. Další řádky se týkají vkládání samotných dat do tabulky. Skript *06_02.php* je zaměřen přímo na zpraco-

vání těchto řádků ze souboru *06_*. Na skriptu žákům demonstrujeme použití daných příkazů při vytváření tabulky v databázi.

Příklad *06_02.php*

```
<HTML><HEAD><TITLE>MySQL</TITLE></HEAD><BODY>
<?
$Conn = mysql_connect ("localhost","jpejsa","rakovnik");
// připojení k databázovému serveru
mysql_select_db ("user_jpejsa_db"); // otevření databáze
$fd = fopen ("./06_", "r"); // otevření souboru
while (!feof ($fd)) // čtení všech řádků ze souboru
    {
    $buffer = fgets($fd, 4096); // celý řádek souboru
    $result = mysql_query($buffer,$Conn);
    // zpracování SQL příkazu, který byl na jednom řádku
    echo $buffer."<BR>"; // vypíše příkaz SQL jazyka
    echo mysql_error(); // vypíše případnou chybu
    }
fclose ($fd); // uzavření souboru
mysql_close ($Conn); // uzavření připojení k serveru MySQL
?>
</BODY></HTML>
```

Tento skript se připojí k databázovému serveru (řádek 3) *localhost* (databázový server se nachází na stejném počítači jako je spuštěn PHP modul) pomocí přihlašovacího jména a hesla. Vybereme s jakou databází budeme pracovat pomocí funkce *mysql_select_db()* a otevřeme soubor, v kterém se nachází samotné příkazy jazyka SQL, příkazem *fopen()*. Pomocí příkazu *while* zařídíme, abychom četli postupně řádek po řádku z daného souboru. Příkazem *echo* vypíšeme přečtený řádek a poté případné chybové hlášení při zpracování MySQL funkce *mysql_query()*. Nakonec uzavřeme otevřený soubor a připojení k databázovému serveru funkcemi *fclose()* a *mysql_close()*. Pokud bude již tabulka vytvořena, vypíše se chybové hlášení při zpracování prvního řádku.

Žáky necháme vytvořit skript, který smaže tabulku *Svatek* v databázi (výsledek naleznete v pracovní aplikaci jako příklad *06_03.php*).

Ve zbývajícím čase (přibližně jedna vyučovací hodina) necháme žáky experimentovat s dalšími příkazy. Dalším námětem pro experimentování je vytvořit databázi obsahující seznam žáků ve třídě s určitými údaji u každého žáka (jméno, příjmení, adresa, telefon, email). Ať zpracují tuto problematiku pomocí formuláře a ošetří ukládání prázdného formuláře. Výsledky dané práce jsou v pracovní aplikaci jako příklady *06_04.php* a *06_05.php*. První z příkladů zobrazuje formulář pro zadávání údajů o žákovi, umožňuje mazat žáka, v případě potřeby vytvoří tabulku v databázi (pokud ještě není vytvořena). Další skript

Tabulka 3.1: Sdružování slov do skupin — zadání

fget	date	strftime	fput	split
file	fclose	eregi	mktime	mkdir
rename	is_dir	localtime	fopen	ereg_replace
ereg	rmdir	time		

Tabulka 3.2: Sdružování slov do skupin — řešení

datum a čas	soubory a adresáře	regulární výrazy
date	fget	split
strftime	fput	eregi
mktime	file	ereg_replace
localtime	fclose	ereg
time	mkdir	
	rename	
	is_dir	
	fopen	
	rmdir	

pouze odstraňuje tabulku i se všemi daty, které v ní jsou (příklad *06_05.php*). Zadání zní takto:

Vytvořte skript (stránky), který bude umět ukládat údaje o žácích ve vaší třídě (jméno, příjmení, adresa, telefonní číslo, email) do databáze. Dále bude muset být vyřešeno mazání údajů z tabulky a výpisy dat z databáze.

3.8 Sedmá sada příkladů — didaktické hry

Rozsah výuky: 2–4 vyučovací hodiny.
Požadované znalosti: znát dané příkazy nebo umět vyhledávat v manuálu PHP.
Znalosti po zvládnutí kapitoly: žáci si upevňují vztahy mezi funkcemi specifickými pro PHP.
Použité metody: metoda didaktické hry, fixační metoda, problémová metoda.

Sdružování slov do skupin je první typ didaktické hry. Vyučující připraví souhrn klíčových slov, příkazů a identifikátorů. Poté je všechny zapíše na tabuli. Žáci mají za úkol seskupit jednotlivé položky do skupin podle toho, jak spolu logicky souvisí. Příkladem je hra z (13) na straně 70–71. Tuto hru můžeme modifikovat na seskupování funkcí PHP. V tabulce 3.1 je uveden příklad takového rozmístění.

Výsledkem jsou tři skupiny, jako v tabulce 3.2. První skupina funkcí pro práci s datem a časem (date, strftime, mktime, localtime, time), druhá skupina pro práci se soubory a adresáři (fget, fput, file, fclose, mkdir, rename, is_dir, fopen, rmdir) a poslední skupinou funkce pracující s regulárními výrazy (split, eregi, ereg_replace, ereg). Po žácích dále požadujeme vysvětlení, na co se používají. Pokud některé nebyly probrány, necháme žáky, aby si sami nastudovali v (1), který je přístupný kdykoliv na internetu[17].

Výměna hodnot dvou proměnných je další typ didaktické hry. Učitel si připraví tři různé nádoby (v našem případě tři sklenice s číselným označením „1", „2" a „3") a dvě odlišné tekutiny (čaj a vodu). Do první sklenice nalijeme čaj a do druhé vodu. Třetí sklenici ponecháme zatím ukrytou před zraky žáků. Zadáme úkol:

„V první sklenici je čaj a v druhé je voda. Přelijte obsah sklenic tak, aby v první byla voda a v druhé byl čaj. Tekutiny se nesmí promíchat!"

Žáci sami po chvíli zjistí, že ke splnění úkolu potřebují ještě jednu sklenici. Po tomto zjištění jim třetí sklenici poskytneme. Necháme si vysvětlit přesný postup přelévání tekutin, který může být následující:

Z první nádoby přeliji čaj do třetí, z druhé vodu do první a z třetí čaj do druhé.

Hru ale můžeme přeformulovat do mnohem náročnější podoby, pokud se omezíme na výměnu dvou přirozených čísel s použitím pouze dvou proměnných. Budeme k tomu potřebovat jen omezený rozsah, který předem stanovíme.

Vyměň obsah dvou proměnných, která obsahují přirozená čísla (včetně nuly) s maximální hodnotou 1000. Nepoužívejte k tomu třetí proměnnou!

Stanovili jsme zde rozsah `0` až `1000`. Žákům po chvíli bádání sdělíme *návod*, jak tento úkol vyřešit. Řekneme jim však jen princip, jakým se to dá řešit. Mějme číslo `12` a `172`. Pokud jedno z nich vynásobíme číslem, které udává přesnost a obě čísla sečteme dostaneme buď `172012` nebo `12172`. Jdou z těchto čísel rozeznat čísla původní? Ano jdou! Teď je již na žácích samotných, jak se s tímto problémem vypořádají.

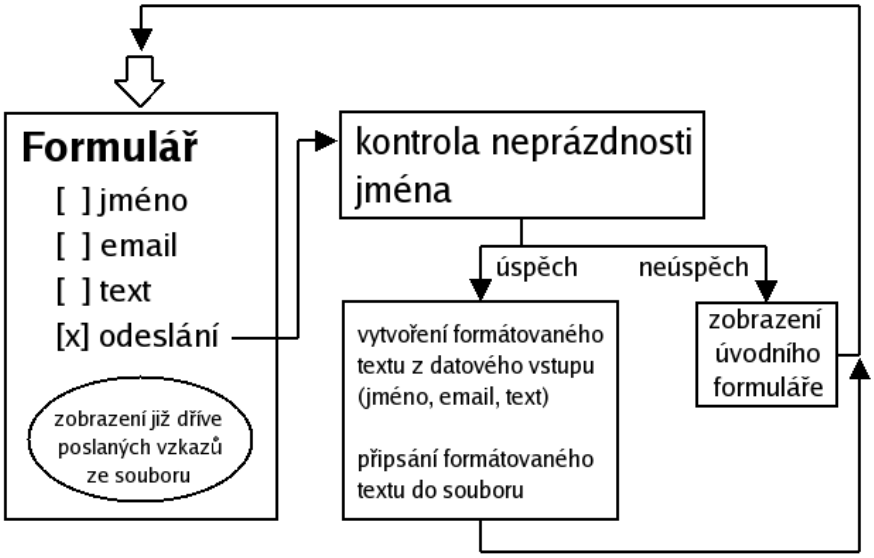
Celý postup se dá zapsat takto:

```
$B = $A+$B*1000;
$A = odděl_desetinná_místa($B/1000);
$B = $B-$A*1000;
```

V PHP skriptu bude zápis velmi podobný, až na funkci *odděl_desetinná_místa()*, která v PHP není. Můžeme si pomoci funkcí na zaokrouhlování *round()*, která má dva parametry. První se zadává číslo, jaké chceme zaokrouhlovat a druhým parametrem je s jakou přesností chceme zaokrouhlovat. Po zadání

[17] Oficiální stránky PHP na <http://www.php.com> nebo na českém zrcadle <http://www.php.cz>

Obrázek 3.8: Grafické znázornění skriptu 05_01.php



přesnosti `0`, se oříznou všechna desetinná místa.

Příklad s podrobným výpisem naleznete v pracovní aplikaci jako příklad *07_01.php*. Je zde ukázána výměna dvou hodnot ve dvou proměnných. Nejdříve je zopakována výměna dvou textů a dvou čísel za pomoci proměnné třetí. Dále následuje hotový skript, který demonstruje výměnu proměnných podle uvedeného zadání. Je zde ukázána výměna proměnné s vyměněnými čísli.

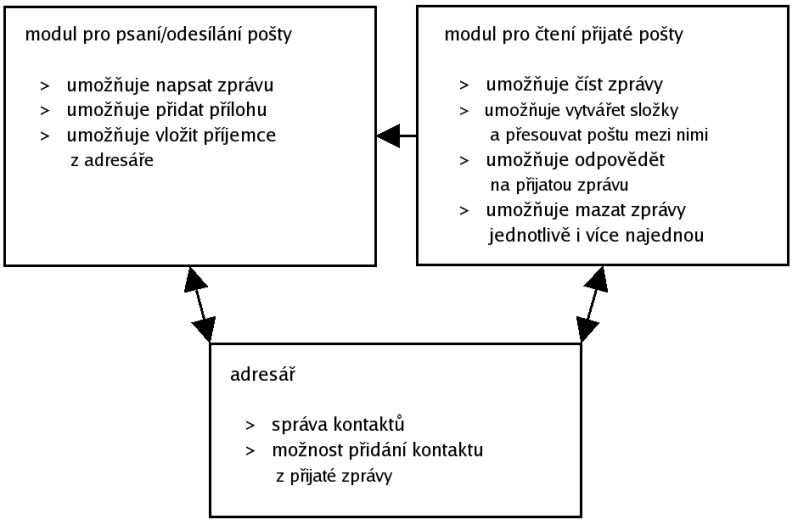
3.9 Osmá sada příkladů — ilustrační metoda

Rozsah výuky: 2–4 vyučovací hodiny.
Požadované znalosti: žádné.
Znalosti po zvládnutí kapitoly: žáci se naučí analyzovat úlohu před jejím samotným řešením.
Použité metody: ilustrační metoda, problémová metoda.

Použití ilustračních metod je velmi široce uplatňováno i v tradičních přístupech k učení v podobě vývojových diagramů, strukturogramů, grafického znázornění dynamických datových struktur a blokové znázornění struktury cyklů. Ilustrační metody znázorním na příkladu diskuse při navrhování rozhraní pro vybírání pošty přes internetové stránky.

Mapa asociací se používá, pokud chceme vytvořit nějaký projekt a ještě nevíme, co se v něm bude konkrétně řešit, kolik na něm bude potřeba autorů a jaké dílčí úseky bude nutno zpracovat. V takovém případě se diskutuje

Obrázek 3.9: Grafické znázornění projektu internetového poštovního klienta



a zapisují se veškeré nápady, které zúčastněné napadnou. Jako příklad uvedu ilustraci řešení příkladu *05_01.php* z kapitoly 3.6.

Učitel nejdříve předvede daný příklad na hotové aplikaci (Kniha návštěv). Dále nakreslí na tabuli, jak skript pracuje. Můžeme využít nákres na obrázku 3.8. Učitel také vysvětlí tento nákres. Po vysvětlení se žáci pustí do práce.

Další námět při využití ilustrační metody je grafické znázornění toho, co bude vše potřeba udělat u širšího projektu. Vezmeme si příklad internetového poštovního klienta. Takový klient musí umět odesílat poštu, a to včetně příloh. Dále musí umět přijímat (v našem případě pouze vybírat) poštu, poštovní schránku ze serveru. Při čtení musí umět správně rozpoznat o jakou přílohu se jedná a nabídnout její správné zobrazení. Určitě by bylo dobré, aby si uživatelé tohoto emailového klienta mohli ukládat kontakty do adresáře — máme tedy další součást projektu.

Učitel by měl využít znalostí žáků a podobné myšlenky nechat volně plynout z úst studentů. Ti totiž budou mít řadu nápadů, co by mohl takový emailový klient všechno obsahovat. Většina studentů již totiž má vlastní zkušenost s takovými klienty a hojně je využívá. Učitel pak již jen bude zapisovat všechny možné nápady na tabuli, z čehož po malé chvíli vznikne mapa projektu. Výhodné je naznačit pomocí šipek všechny různé vazby mezi jednotlivými částmi. Příklad mapy projektu si můžete prohlédnout na obrázku 3.9. Můžete si všimnout vzájemného propojení propojení adresáře mezi modulem pro psaní i odesílání pošty (při psaní i čtení pošty vzájemná spolupráce s adresářem) a dále vazbu mezi čtením a psaním zprávy (možnost odpovědi na zprávu).

Kapitola 4

Zpracování WWW stránek s příklady

Pro lepší využití všech příkladů jsem zpracoval internetové stránky s možností přímého spuštění všech příkladů a následné možnosti modifikace stávajících či vytváření nových skriptů. V dalším textu budu používat označení „pracovní aplikace", jelikož s těmito stránkami budete pracovat v průběhu další kapitoly. Ještě poznamenám, že pracovní aplikace je odzkoušena v prohlížečích *Internet Explorer 6.0* a *Mozilla 1.4*. Stránky by se měly správně zobrazovat ve většině internetových prohlížečů.

4.1 Instalace WWW stránek

Stránky obsahující celou diplomovou práci ke shlédnutí a k ní pracovní aplikaci naleznete na internetu na adrese http://home.zcu.cz/~jpejsa/diplomka/. Obsah těchto stránek (nejedná se o uvedenou internetovou adresu) si může kdokoliv zpřístupnit na internetu nebo na svém počítači, přičemž stačí vzít přiložený CD–ROM k této diplomové práci a překopírovat obsah adresář *www_stranky*.

Existují tedy dvě možnosti, jak si tyto stránky zprovoznit i po té, co již nebudou na již zmíněné adrese. První možnost počítá s tím, že máte na nějakém internetovém serveru konto a možnost nahrávat vlastní HTML stránky. Důležitou podmínkou je, aby tento server podporoval skriptovací jazyk PHP. Pokud chcete využívat přednosti databáze MySQL, musíte mít rovněž k dispozici takovou podporu, nejlépe přímo na serveru. Poté stačí jednoduše překopírovat adresář *diplomka*, který se nachází na přiloženém CD–ROMu, přímo na server, kde budou umístěny tyto stránky. Od této chvíle by mělo být vše funkční[1].

[1]Jedním takovým serverem je *http://www.webzdarma.cz*.

Druhou možností je vytvořit si vlastní server. Budeme k tomu potřebovat HTTP server, který umožňuje přidat podporu PHP. Takový serverem je Apache, který přímo podporuje vložení PHP jako modulu. Návod, jak tento server nainstalovat, naleznete v podrobné dokumentaci u každé distribuce. Pokud budeme chtít používat na našem počítači i databázi MySQL, musíme si ji také nainstalovat. Návod nalezneme rovněž přímo v distribuci dané aplikace. Všem doporučuji první možnost zprovoznění stránek z důvodu jednoduchosti.

Po instalaci nesmíme zapomenout na nastavení přístupových práv u souborů v podadresáři *priklady*. Pokud bychom nenastavili práva serveru Apache číst a zároveň zapisovat do těchto souborů, vypisovala by pracovní aplikace chybová hlášení a nepracovala by správně, jelikož by neměla přístup na dané soubory. Nastavení se liší podle zvoleného systému (UNIX, Windows). V případě uživatelů systému Windows (souborový systém FAT) není potřeba žádného nastavení, jelikož souborový systém neumožňuje nastavovat žádná přístupová práva. Uživatelé UNIXu nastaví práva příkazem *chmod*.

Pro správnou funkci všech skriptů je potřeba nastavit práva pro zápis na tyto soubory:

```
0.php
02_
03_
03__
```

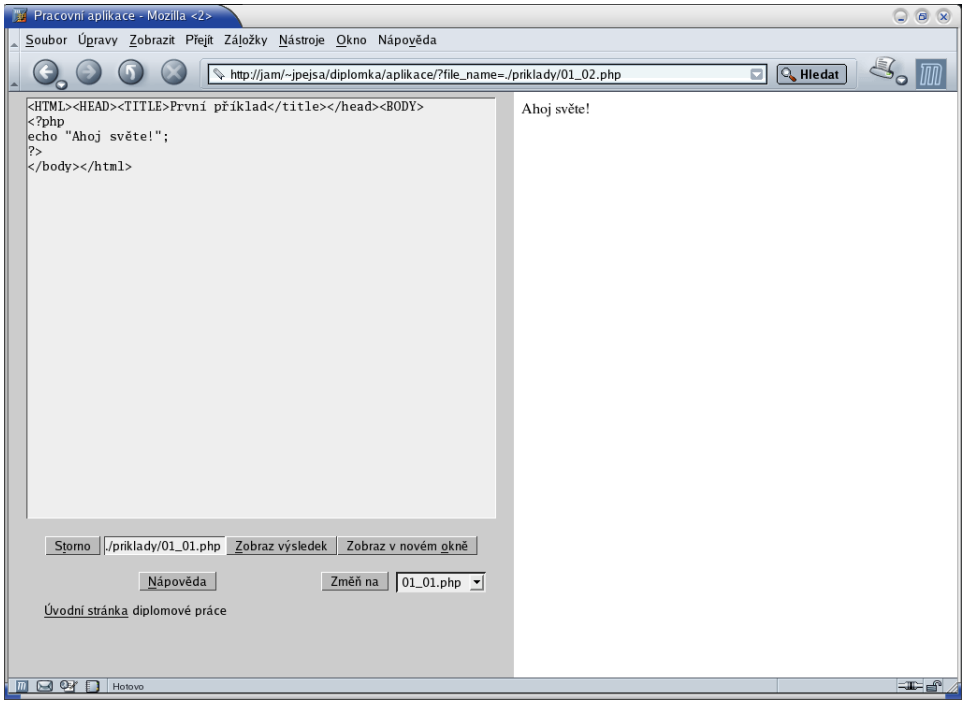
4.2 Spuštění stránek pracovní aplikace

Spuštěním se rozumí otevření stránek diplomové práce (respektive pracovní aplikace) v prohlížeči určeného pro prohlížení internetových stránek. Jedno možné umístění je k dispozici v kapitole 4.1 na předchozí straně. Příkladem takových prohlížečů jsou Internet Explorer nebo Mozilla. Pokud si otevřete tyto stránky, můžete již studovat všechny příklady uvedené v kapitole 3. V následující kapitole se dozvíte, jak se pracovní aplikace ovládá.

4.3 Popis ovládání WWW stránky s příklady

Pracovní aplikace (obrázek 4.1) je rozdělena na dva rámce. Levá strana obsahuje zdrojový kód skriptu (obsah souboru uloženého na serveru) a možnosti práce se skriptem jako jsou ukládání a otevírání jiných skriptů přímo na serveru. Pravá část ukazuje výsledný soubor po zpracování serverem (již to, co je posláno klientovi). Více o zpracování skriptu serverem najdete v kapitole 3.1.

Obrázek 4.1: Pracovní aplikace



Na obrázku 4.1 vidíte výsledné zobrazení aplikace. V levé části vidíte zdrojový kód skriptu PHP, příklad *„01_01.php”* ze strany 3.2, a v pravé části zobrazení přeloženého souboru v prohlížeči. Navíc jsem si nechal zobrazit zdrojový kód již přeloženého skriptu, můžeme tedy porovnat, co interpret PHP zpracoval a co ponechal beze změny.

Ovládání je navrženo tak, aby bylo co nejjednodušší. Tlačítko *Storno* vrací původní obsah formuláře. Tlačítko *Ulož* ukládá zdrojový kód do souboru, jehož jméno je zobrazeno vedle tlačítka. Z bezpečnostních důvodů je povoleno zapisovat pouze do souboru *„0.php”*, proto se při jiných otevřených souborech toto tlačítko ani nezobrazuje. Tlačítkem *Zobraz výsledek* znovu načteme obsah pravé části rámce. Tlačítko *Zobraz v novém okně* využijeme, pokud budeme chtít zobrazit výsledný skript v novém okně prohlížeče. K dispozici je i nápověda, kterou vyvoláme tlačítkem *Nápověda*. Abychom mohli načítat jiné příklady, použijeme roletové menu (vybereme příklad – soubor) a stiskneme tlačítko *Změň na*. Poté se ihned nahraje námi zvolený příklad. Podrobně popsané příklady nalezneme v kapitole 3 na straně 30.

Pokud se rozhodneme experimentovat s některým příkladem, použijeme

tento jednoduchý postup[2]. Po rozhodnutí s jakým příkladem budeme chtít pracovat, se přepneme do zdrojového kódu skriptu PHP (levý rámec pracovní aplikace). Celý zdrojový kód označíme (klávesová zkratka Ctrl + A) a zkopírujeme do schránky (Ctrl + C). Změníme otevřený soubor na *0.php* (pomocí roletového menu a tlačítka *Změň na*). Mělo by se zobrazit tlačítko *Ulož*. V tomto souboru (zdrojovém kódu) znovu označíme celý kód a smažeme (klávesa *Delete*). Poté vložíme námi kopírovaný skript (Ctrl + V) a můžeme s ním již pracovat.

Při práci s pracovní aplikací (s některým příkladem) se může stát, že se mohou vypisovat chybová hlášení. Ty jsou vypisována přímo PHP interpretem a informují nás, kde máme chybu a jakého typu je tato chyba. Nejčastěji se jedná o chybnou syntaxi.

4.4 Popis pracovní aplikace

Pracovní aplikaci jsem vytvořil pomocí znalosti značkovacího jazyka HTML, kaskádových stylů CSS, Java Scriptu a samozřejmě skriptovacího jazyka PHP.

- *index.php*
  Jedná se výchozí skript pro dané umístění na internetu (např. http://home.zcu.cz/~jpejsa/diplomka/ nebo http://jpejsa.wz.cz/diplomka/). Určuje rozdělení rámců[3] a chování celé aplikace. Pravý rámec zobrazuje zpracovaný příklad nebo nápovědu. Levý rámec zobrazuje textové pole a další ovládací prvky. Pokud se jedná o příkladový soubor *0.php*, umožňuje ho uložit přímo na server. Samotný skript znemožňuje zápis do jakéhokoliv jiného souboru, než do mnou určeného (zabezpečení proti neoprávněnému přepsání jiných souborů). Aby mohl být samotný soubor *0.php* přepisovatelný, musí být na něj na serveru nastavena práva pro zápis. Dále zobrazuje tlačítka formuláře pro práci s příklady.

- *config.php*
  V tomto souboru můžeme nastavit doladění pracovní aplikace (šířka rámce v prohlížeči, výška a šířka textového pole se zobrazeným zdrojovým kódem skriptu).

- adresář *priklady*
  Obsahuje všechny příklady, které jsou popsány v kapitole 3. Většina zdrojových kódů příkladů je k dispozici v téže kapitole. Ostatní zdrojové

[2]Do příkladového materiálu není dovoleno zapisovat, vše je uloženo na serveru.
[3]Značka *FRAMESET* a *FRAME*, více najdete v (12).

kódy naleznete na přiloženém CD–ROMu nebo v pracovní aplikaci na internetu. Všechny příklady je možno kdykoliv spustit (prohlížet) pomocí pracovní aplikace.

Výpisy zdrojových kódů těchto skriptů naleznete v dodatku A.

4.5 Shrnutí příkladů a souborů obsažených v pracovní aplikaci

01_01.php, 01_02.php — výpisy na „obrazovku”
01_03.php, 01_04.php, 01_05.php — práce s proměnnými
01_03.php, 01_06.php — aritmetické operace
01_07.php, 01_08.php — aritmetické operace pomocí formuláře
02_01.php, 02_02.php — počítadlo přístupů
03_01.php, 03_02.php, 03_03.php — počítadlo přístupů
04_01.php, 04_02.php, 04_03.php – práce s polem, tvorba vlastních funkcí
04_04.php — průchod daným adresářem a zobrazení náhledů obrázků v něm obsažených
05_01.php, 05_02.php — kniha návštěv
05_03.php, 05_04.php — odesílání emailu, odesílání SMS na mobilní telefon
06_01.php, 06_02.php, 06_03.php — zobrazení svátku na dnešní den, propojení s databází MySQL
06_04.php, 06_05.php — evidence žáků pomocí MySQL (jméno, příjmení, atd.)
07_01.php — výměna obsahu dvou proměnných

0.php — soubor určený primárně pro zápis (možnost psaní vlastního skriptu)
*02_* — ukládání počtů přístupů skripty *02_01.php* a *02_02.php*
*03_* — ukládání počtů přístupů skripty *03_01.php až 03_03.php*
03__ — ukládání IP adresy skriptem *03_03.php*
*06_* — příkazy MySQL pro skript *06_02.php*
06__ — zkrácená verze souboru *06_*

Kapitola 5

Závěr

Komunikativní výuka skriptovacího jazyka PHP přináší nové pojetí učiva a navrhuje alternativní metody, oproti dosavadnímu způsobu vyučování. Na rozdíl od tradičního přístupu výuky se nejvíce odráží v rozšíření metodických postupů, ve zvýšeném zastoupení formativní stránky učiva (tj. dovedností žáků) a v celkové změně pohledu na úlohu žáků a vyučujících. Komunikativní metody vyžadují od žáků větší samostatnost a aktivní spoluúčast na řešení daných problémů při výuce. Dále přibližují charakter výuky reálnému životu a tím napomáhají většímu uplatnění žáků ve společnosti.

Praktická implementace komunikativní výuky klade vyšší nároky na učitele než tradiční přístupy. Nejvíce tak v oblasti učebních materiálů, které je často nutné vytvářet zcela nové (jedná se o přípravu činností, které tradiční metody nevyužívají). Některé zdroje materiálů jsou naopak snadno přizpůsobitelné a jejich využití snižuje časovou náročnost přípravy.

Diplomová práce v kapitole 2 zkoumá možnosti využití komunikativního přístupu k výuce skriptovacího jazyka PHP, přičemž se vychází z komunikativní výuky cizích jazyků. Dále jsou popsány specifika výuky programování na střední škole a zvláštnosti, které se vyskytují při výuce programovacího jazyka oproti cizímu jazyku. Na konec kapitoli jsem zařadil nahlédnutí na otázku hodnocení jakožto součásti výuky.

V kapitole 3 jsou již jednotlivé metody ilustrovány na konkrétních příkladech. Snažil jsem se, aby čtenář měl možnost si vše vyzkoušet (při připojení na internet). Tím se dosáhne lepších výsledků, jelikož čtenář pouze nečte, ale také pracuje s počítačem a se skriptovacím jazykem PHP. Tímto tedy otevírám možnost samostatné výuky jednotlivce. Diplomová práce by měla pomáhat učiteli, který se zde může dozvědět, jakým jiným způsobem může naučit danou látku. Učitel na samotné příklady spoléhat nemůže, jelikož diplomová práce nebyla tvořena jako celistvý výukový materiál. Ta slouží jako ilustrace jednotlivých metod komunikativní výuky a není tedy metodickým návodem pro-

gramování v PHP. Můžeme využít všechny příklady uvedené v diplomové práci, avšak není zde vysvětleno zdaleka vše, co se skriptovacího jazyka PHP týká. Zájemce o samotný jazyk PHP proto odkazuji na literaturu, ve které naleznou veškeré informace týkající se skriptovacího jazyka (1; 5).

Věřím, že diplomová práce najde uplatnění nejen u učitelů na středních školách, ale i u jednotlivců.

Literatura

[1] Bakken, Schmid Manuál PHP : PHP Documentation Group, 2002. Dostupné na <http://www.php.cz>.

[2] Billows, L. F. Kooperativní technika vyučování cizímu jazyku 2. vyd. Plzeň : Západočeská univerzita, 1995. 203 s. ISBN 80–7043–173–3.

[3] Brumfit, C., J., Johnson, K. The Communicative Approach to Language Teaching. 1. vyd. Oxford : Oxford University Press, 1979. 243 s. ISBN 0–19–437078.

[4] Canale, M. a Swain M. Theoretical bases of communicative approaches to second language teching and testing, 1. vyd. : Applied Linguistics Vol. 1. No. 1 strany 1–47, 1980.

[5] Castagnetto, Rawat, Schumann, Scollo, Veliath PHP programujeme profesionálně 1. vyd. Praha : Computer Press, 2001. 656 s. ISBN 80–7226–310–2.

[6] Číhalová, E. a Mayer, I. Jak rozvíjet komunikaci a spolupráci. Zkušenosti z praxe na 1. st. ZŠ Praha : Agentura STROM.

[7] Choděra, R. Moderní výuka cizích jazyků 1. vyd. Praha : APRA, 1993. 135 s.

[8] Kosek, J. PHP, tvorba interaktivních internetových aplikací – podrobný průvodce 1. vyd. Praha : Grada Publishing, 1998. 492 s. ISBN 80–7169–373–1.

[9] Littlewood, W. Comunicative Language Teaching, An Introduction 13. vyd. Cambridge : Cambridge University Press, 1991. 108 s. ISBN 0–521–28154–7.

[10] Miller, Z., články autora publikované na internetu : Dostupné na <http://miller.wz.cz/>.

[11] Pecinovský, Rudolf, články autora publikované na internetu : Dostupné na <http://www.ceskaskola.cz>.

[12] Raggett D. HTML Reference Manual : W3C 2002. Dostupné na <http://www.w3c.org/school/>.

[13] Vrbík, V. Komunikativní přístup k výuce programovacího jazyka Plzeň : Západočeská univerzita 1995. 150 s. ISBN 80–7082–856–0.

[14] Widdowson, H. G. Teaching Language as Communication 1. vyd. Oxford : Oxford University Press 1978.

[15] Widenius M., Axmark D. MySQL Reference Manual : O'Reilly & Associates 2002, ISBN 0–593–00265–3.

Příloha A

Zdrojové kódy pracovní aplikace

Soubor *index.php*

```
<?php
// ============================================================
// pracovní aplikace
// ============================================================
include "./config.php";
// ============================================================
// zobrazí nápovědu
function napoveda()
{
echo "<h1>Nápověda</h1>";
echo "</body></html>";
exit;
}
// ============================================================
// ramce() zobrazí stránku s definicí rámců pracovní aplikace
// levá strana...editace_prikladu
// pravá strana...ukazka_prikladu
function ramce()
{
global $file_name,$frame_col1;
$title="Pracovní aplikace";
$style="./../style.css";
include "./../head.php";
echo "<FRAMESET COLS=\"".$frame_col1.",*\" FRAMESPACING=0 ";
echo " FRAMEBORDER=no SCROLLING=AUTO>\n";
echo "<FRAME SRC=\"./?editace_prikladu=1&file_name=".$file_name."\"";
echo " ALIGN=top NAME=editace_prikladu MARGINHEIGHT=0 ";
echo " FRAMEBORDER=yes SCROLLING=AUTO>\n";
echo "<FRAME SRC=\"".$file_name."\" ALIGN=top ";
echo " NAME=ukazka_prikladu FRAMEBORDER=yes SCROLLING=AUTO>";
echo "</FRAMESET>\n";
echo "<noframes>\n";
echo "Váš prohlížeč nepodporuje zobrazování rámců,\n";
echo "použijte jiný, například <a href=";
echo "\"http://www.mozilla.org\" target=\"_blank\">Mozilla</a>.\n";
echo "</noframes>\n";
}
// ============================================================
// vypsání chyby o neuložení do souboru
function chyba_pri_zapisu_do_souboru()
```

```
{
echo "<h1>Chyba</h1>";
echo "<p>Při zapisování do souboru <b>./priklady/0.php</b> ";
echo "nastala chyba.<br>Zkontrolujte, zda má server právo ";
echo "zápisu do tohoto souboru. ";
echo "<p><a href=./ target=_top>Pracovní aplikace</a>";
echo "</body></html>";
exit;
}
// ============================================================
// ukládání do souboru!!!
function uloz_do_souboru()
{
global $file_name, $source_file;
if (!isset($file_name)) $file_name="./priklady/0.php";
//echo "ahoj<br>".$source_file.$POST_VARS;
if (isset($source_file))
 {
 if ($file_name=="./priklady/0.php")
  if (is_writable($file_name))
   {
   @$file = fopen ("./$file_name", "w");
   @fputs($file, $source_file);
   @fclose($file);
   header ("Location: ./?reload=1&file_name=$file_name");
   //header ("Location: ./?file_name=$file_name");
   exit;
   }
  else
   {
   header ("Location: ./?error=1");
   exit;
   }
 }
}
// ============================================================
// editace_prikladu() zobrazuje stránku na levé straně rámce
function editace_prikladu()
{
global $file_name,$ta_row,$ta_cols;
$title="Editace zdrojového kódu příkladu";
$style="./../style.css";
include "./../head.php";
echo "<BODY><center>";
echo "<TABLE align=center BORDER=0 CELLSPACING=0 CELLPADDING=0";
echo " WIDTH=".($frame_col1-10).">\n";
echo "<TR>\n<TD ALIGN=center COLSPAN=3>\n";
// formulář pro ukládání
// (zobrazení zdrojového kódu skriptu + tlačítka Storno a Ulož)
echo "<FORM METHOD=POST target=editace_prikladu name=hlavni";
echo " ACTION=\"./?editace_prikladu=1&ulozit=1&file_name=";
echo $file_name."\">\n";
echo "<TEXTAREA class=policko ALIGN=texttop NAME=source_file";
echo " ROWS=$ta_row COLS=$ta_cols wrap=off>\n";
  // vypsání zdrojového kódu skriptu do HTML stránky
  $file = fopen ("./$file_name", "r"); // otevřu pro čtení
  while (!feof ($file))
    { $buffer = fgets($file, 4096); echo $buffer; }
```

```
  fclose($file);
echo "</TEXTAREA>\n";
echo "</TR>\n</TD>\n";
echo "</table>\n";
// prázdná mezera
echo "<TABLE align=center BORDER=0 CELLSPACING=0 CELLPADDING=0";
echo " WIDTH=".($frame_col1-10).">\n";
echo "<tr><td>";
echo "<img src=./images/bod_pruhledny.gif border=0 height=5>";
echo "</td></tr></table>";
// ovládací tlačítka...
echo "<TABLE align=center BORDER=0 CELLSPACING=0 CELLPADDING=0";
echo " WIDTH=".($frame_col1-10).">\n";
echo " <TR>\n";
echo "<TD ALIGN=right valign=top>\n";
// reset tlačítko
echo "<button type=reset TITLE=\"Zruš editaci\" class=tlacitko";
echo " accesskey=T>S<u>t</u>orno</button>\n";
echo "</TD>\n<TD ALIGN=right valign=top>\n";
// tlačítko pro ukládání (povolení/zakázání zobrazení)
if ($file_name=="./priklady/0.php")
  {
  echo "<button class=tlacitko align=center name=potvrd";
  echo " type=submit accesskey=U title=\"Uloží soubor na serveru\"";
  echo "><u>U</u>lož</button>\n";
  }
echo " \n</TD>\n<TD ALIGN=right valign=top>\n";
// zobrazí jméno právě zobrazeného skriptu
echo "<INPUT class=policko TITLE=\"Právě zobrazovaný soubor\"";
echo " TYPE=TEXT size=17";
echo " NAME=file MAXLENGTH=20 READONLY value=\"".$file_name."\">\n";
echo "</TD>\n<TD ALIGN=right valign=top>\n</form>\n";
// zobrazení (znovunačtení) výsledku
echo "<form action=$file_name target=ukazka_prikladu name=reload>\n";
echo "<BUTTON class=tlacitko align=center type=submit";
echo " title=\"Zobrazí výsledný skript v pravém rámci\" accesskey=Z";
echo "><u>Z</u>obraz výsledek</BUTTON>\n</form>\n";
echo "</td>\n<td ALIGN=right valign=top>\n";
echo "<form action=$file_name target=_blank name=new_window>\n";
echo "<button class=tlacitko accesskey=O type=submit";
echo " title=\"Zobrazí výsledný skript v novém okně\">";
echo "Zobraz v novém <u>o</u>kně</button>\n";
echo "</form>\n";
echo "</TD>\n</TR>\n</table>\n";
// prázdná mezera
echo "<TABLE align=center BORDER=0 CELLSPACING=0 CELLPADDING=0";
echo " WIDTH=".($frame_col1-10).">\n";
echo "<tr><td>";
echo "<img src=./images/bod_pruhledny.gif border=0 height=5>";
echo "</td></tr></table>";
// další tlačítka...
echo "<TABLE align=center BORDER=0 CELLSPACING=0 CELLPADDING=0";
echo " WIDTH=".($frame_col1-10).">\n";
echo "<TR>";
echo "   <TD ALIGN=right width=40% valign=top>";
// tlačítko nápovědy
echo "<form action=./?napoveda=1 target=ukazka_prikladu name=help>";
echo "<BUTTON class=tlacitko align=center name=napoveda accesskey=N";
```

```
echo " type=submit title=\"Zobrazí v pravém rámci nápovědu\">";
echo "<u>N</u>ápověda</BUTTON>\n</form>\n";
echo "</TD>\n<TD ALIGN=right width=40% valign=top>\n";
// změna na jiný skript
echo "<FORM NAME=change METHOD=POST target=_top";
echo " ACTION=\"./?file_name=$file_name\">";
echo "<button type=submit class=tlacitko name=change_to";
echo " title=\"Změní na vybraný soubor\">Změň na</button>\n ";
echo "</TD>\n<TD ALIGN=left valign=top>\n";
echo "<SELECT class=policko NAME=file_name";
echo " title=\"Změní na vybraný soubor\"";
echo " onchange=\"this.form.submit()\"";
echo ">";
   // vybereme v adresáři ./priklady všechny *.php soubory
   $adresar = opendir("./priklady");
     while ($soubor=readdir($adresar))
        {
        if (($soubor!=".") & ($soubor!=".."))
          if (eregi(".php",$soubor))
            {
            echo "<option value=\"./priklady/$soubor\"";
            if ("./priklady/".$soubor==$file_name) echo " selected ";
            echo ">$soubor</option>";
            }
        }
echo "   </SELECT>\n";
echo "  </FORM>\n";
echo "</TD>\n</TR>\n";
echo "<TR>\n<TD ALIGN=left COLSPAN=3>\n";
echo "  <A HREF=\"./../\" TARGET=_top>Úvodní stránka</A>";
echo " diplomové práce";
echo "</TD>\n</TR>\n";
echo "</TABLE>\n</center>\n";
echo "</BODY>\n";
}

// ============================================================
// hlavní program
// ============================================================
if (isset($reload))
     {
     echo "<html><head></head><body onload=\"";
     echo "javascript:top.document.location=";
     echo "'./?filename=$file_name'\"></body></html>";
     }
if (isset($source_file)) uloz_do_souboru();
if (isset($napoveda))
     { $style="./../style.css";
       include "../head.php"; napoveda(); exit; }
if (isset($error))
     { $style="./../style.css";
       include "../head.php"; chyba_pri_zapisu_do_souboru(); exit; }
if (isset($editace_prikladu)) editace_prikladu();
if (!isset($editace_prikladu)) ramce();

echo "</HTML>\n";
?>
```

Soubor *config.php*

```
<?php
// ============================================================
// config.php nastavení pracovní aplikace
// ============================================================
if (get_magic_quotes_gpc()) {
// převezme chování GPC (GetPostCookies) proměnných
for (reset($_GET); list($k, $v) =
    each($_GET); ) $$k = stripslashes($v);
for (reset($_POST); list($k, $v) =
    each($_POST); ) $$k = stripslashes($v);
for (reset($_COOKIE); list($k, $v) =
    each($_COOKIE); ) $$k = stripslashes($v); }
// ============================================================
// globální možnosti nastavení pracovní aplikace
// ============================================================
$frame_col1=540; // velikost levého rámce
$ini_soubor="./priklady/0.php";
$ta_row=27;
$ta_cols=60;


// ============================================================
// další nastavení - proměnná file_name a source_file
if (isset($POST_VARS["file_name"]))
                $file_name=$_POST["file_name"];
if (isset($GET_VARS["file_name"]))
                $file_name=$_GET["file_name"];
if (!isset($file_name))
                $file_name=$ini_soubor;
if (isset($POST_VARS["source_file"]))
                $source_file=$_POST["source_file"];
?>
```

Příloha B

Zdrojové kódy stránek diplomové práce

Soubor *index.php*

```
<?php
$title="Jan Pejša - Diplomová práce";
$style="./style.css";
include "./head.php";
if (isset($_GET["str"]))
   {
   include "./".$_GET["str"];
   exit;
   }
?>
<BODY>
<center>
<table width="100%" cellspacing="0" border="0" cellpadding="0"
align="center" height="550">
<tr><td width="100%" height="100%" align="center" valign="center">
<TABLE BORDER="0" CELLSPACING="0" CELLPADDING="0" WIDTH="800"
ALIGN="center">
  <TR>
    <TD WIDTH="10"> </TD>
    <TD align="left" valign="top" width="180">

<H2> <br> </H2>

<b>Stručný obsah</b>
<ul>
  <li> <a href="./diplomka.pdf">Diplomová práce</a> (formát pdf)
  <li> <a href="./aplikace/">Pracovní aplikace</a> (zpracované příklady)
</ul>

    </TD>
    <TD WIDTH="40"> </TD>
    <TD align="left" valign="top">

<H2>DIPLOMOVÁ PRÁCE<br>Jan Pejša</H2>

<P ALIGN="justify">
Diplomovou práci jsem začal psát 22. dubna 2002.
```

```
Více se jí zabýval však až v lednu 2003.
Přepracovávat jsem jí začal 24. května 2002.
Samotnou práci píši v editoru <i>L<sub>Y</sub>X</i>, což je
nadstavba <i>L<sub>A</sub>T<sub>E</sub>Xu</i>.
Pro tvorbu HTML stránek používám editor <i>Quanta</i>.
Obrázky kreslím v grafickém editoru <i>GIMP</i>, kvalitu a formát
měním pomocí prohlížeče/editoru <i>XnView</i>.

<P ALIGN="justify">
<A HREF="./aplikace/">Pracovní aplikace</A>, kde si můžete
vyzkoušet příklady použité v diplomové
práci, můžete zde také napsat jakýkoliv PHP skript a ihned
vidět výsledek (vše ve vašem
prohlížeči).  Tuto aplikaci jsem optimalizoval pro rozlišení
1024x768, proto se omlouvám těm,
co používají menší.

<P ALIGN="justify">
Text vlastní diplomové práce ve formátu <A HREF="./diplomka.pdf">PDF</A>.
<a href="./?str=harmonogram.html">Harmonogram</a>
a <a href="./?str=zadani.html">zadání</a> práce.

    </TD>
    <TD WIDTH="10"> </TD>
  </TR>
</TABLE>
</td></tr></table>
</center>
</BODY>
</HTML>
```

Soubor *head.php*

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<HTML>
  <HEAD>
    <META HTTP-EQUIV="Content-Type" CONTENT="text/html;charset=ISO-8859-2">
    <META HTTP-EQUIV="Cache-control" content="no-cache">
    <META HTTP-EQUIV="pragma" content="no-cache">
    <TITLE><? echo $title; ?></TITLE>
    <LINK href="<? echo $style; ?>" rel=stylesheet type=text/css>
  </HEAD>
```

Soubor *style.css*

```
BODY {
        FONT-FAMILY:  Verdana, Tahoma, sans-serif, Arial;
        FONT-SIZE: 10pt;
        COLOR: #000000;
        BACKGROUND-COLOR: #CCCCCC;
        MARGIN: 0px;
        PADDING-BOTTOM: 5px;
        PADDING-LEFT: 5px;
        PADDING-RIGHT: 5px;
        PADDING-TOP: 5px;
```

```
}
A {
        text-decoration:underline;
        COLOR: #000000;
}
A:hover {
        text-decoration:none;
        COLOR: #000000;
        BACKGROUND-COLOR: #FFFFFF;
}
.tlacitko {
        FONT-SIZE: 10pt;
        COLOR: #000000;
        border-top: #FFFFFF thin solid;
        border-left: #FFFFFF thin solid;
        border-right: #444444 thin solid;
        border-bottom: #444444 thin solid;
        BACKGROUND-COLOR: #BBBBBB;
  }
.tlacitko:hover {
        COLOR:#000000;
        BACKGROUND-COLOR: #FFFFFF;
  }
.policko {
        FONT-SIZE: 10pt;
        COLOR: #000000;
        border-top: #444444 thin solid;
        border-left: #444444 thin solid;
        border-right: #FFFFFF thin solid;
        border-bottom: #FFFFFF thin solid;
        BACKGROUND-COLOR: #EEEEEE;
  }
```

Příloha C

Přiložený CD–ROM

K diplomové práci je přiložen CD–ROM, které obsahuje všechny zdrojové kódy skriptů a stránek použitých v diplomové práci (včetně kompletních stránek pracovní aplikace). Diplomová práce je k dispozici i v elektronické podobě, ve formátu PDF a LyX.

LyX je nadstavba LaTeXu, který umožňuje pohodlnou tvorbu publikací. LaTeX je distribuce TeXu, sázecího programu pro profesionální vzhled dokumentů. Prohlížeč na PDF dokumenty je přiložen (Adobe Acrobat Reader). Editaci stránek a skriptů provádím pomocí programu pro internetovou publikaci *quanta*. Grafické objekty jsem vytvořil pomocí programů *dia*, *gimp* a *xnview*. Program Mozilla je volně šiřitelný prohlížeč internetových stránek.

Na CD–ROMu se nachází:

- *diplomová práce (formát PDF a LyX),*
- *zdrojové kódy pracovní aplikace včetně příkladů použitých v diplomové práci,*
- *internetový prohlížeč mozilla (verze pro windows),*
- *Adobe Acrobat Reader (verze pro windows).*

Evidenční list

Souhlasím s tím, aby moje diplomová práce byla půjčována k prezenčnímu studiu v Univerzitní knihovně ZČU v Plzni.

V Plzni dne 5. ledna 2003 Jan Pejša

Uživatel stvrzuje svým čitelným podpisem, že tuto diplomovou práci použil ke studijním účelům a prohlašuje, že ji uvede mezi použitými prameny.

Jméno a příjmení	**Fakulta/katedra**	**Datum**	**Podpis**