Česká zemědělská univerzita v Praze

Provozně ekonomická fakulta

BAKALÁŘSKÁ PRÁCE

Tomáš Chvátal

Programovací a skriptovací jazyky v prostředí WWW

Katedra Informačních Technologií

Ing. Pavel Šimek, Katedra Informačních Technologií

Studijní program: Informatika

© 2008

Těmito pár slovy děkuji vedoucímu mé bakalářské práce Ing. Pavlu Šimkovi za jeho odbornou pomoc, konzultace a pomoc při korektuře. Dále bych rád poděkoval slečně Anně Janáčkové za vytvoření testovacích skriptů z pohledu programátora-začátečníka.

Prohlašuji, že jsem svou bakalářskou práci napsal(a) samostatně a výhradně s použitím citovaných pramenů. Souhlasím se zapůjčováním práce a jejím zveřejňováním v souladu s licencí Creative Commons Attribution-Share Alike.

V Praze dne 28. dubna 2008 Tomáš Chvátal

Programovací a skriptovací jazyky v prostředí WWW

Programming and scripting languages for WWW

Anotace

Bakalářská práce se zabývá popisem vybraných oblíbených programovacích a skriptovacích jazyků určených pro webové prezentace, jejich historií a plánovaným vývojem. Pokouší se shrnout jejich výhody a nevýhody pro užití na firemní prezentaci tvořenou vlastním IT (informační technologie) týmem. Srovnává vybrané programovací a skriptovací jazyky. Porovnává dobu potřebnou k jejich vytvoření programátorem a dobu jejich běhu na PC.

Anotation

Presented thesis deals with description of various popular programing and scripting languages intended for web presentation, theirs history and planes for further development. Attempts to sumarize their disavantages and benefits for usage on corporate presentation created by own IT (information technologies) team. Compare selected programing and scripting languages. Compare time needed for their creation by programmer and runtime on PC.

Klíčová slova

Apache, C/C++, IIS, Internet, Java, Lighttpd, PHP, Python, Perl, Popis skriptovacích a programovacích jazyků, Porovnání skriptovacích a programovacích jazyků, Ruby.

Keywords

Apache, C/C++, Comparsion of scripting a programming languages, Description of scripting a programming languages, IIS, Internet, Java, Lighttpd, PHP, Python, Perl, Ruby.

Obsah

0 Úvod **5**

1 Cíl práce a metodika **7**

2 Programovací jazyky pro WWW **9**

2.1 C, C++ 10

2.1.1 Shrnutí fundamentálních vlastností jazyka C 12

2.1.2 Historie 13

2.2 Java 15

2.2.1 Shrnutí výhod jazyka Java 16

2.2.2 Historie 17

2.2.2.1 Oak 17

2.2.2.2 Star 7 18

2.2.2.3 FirstPerson 18

2.2.2.4 Java a WWW 19

3 Skriptovací jazyky pro WWW **20**

3.1 Perl 21

3.1.1 Klíčové znaky jazyka Perl 22

3.1.2 Historie 23
3.1.2.1 Název 24
3.2 PHP 24
3.2.1 Historie 25
3.3 Python 26
3.3.1 Historie 28
3.4 Ruby 29
3.4.1 Historie 30

4 Pracovní a vývojová prostředí 31
4.1 Web-servery 31
4.1.1 Apache 32
4.1.2 Lighttpd 32
4.2 Vývojová prostředí 33
4.2.1 Vim 33
4.2.2 Kate 35

5 Porovnání programovacích a skriptovacích jazyků 36
5.1 Java 37
5.2 PHP 38
5.3 Python 39
5.4 Ruby 40
5.5 Tabulka časů 41

6 Závěr 42

A Přílohy VI

0 Úvod

Internetové služby každým rokem stávají stále více protěžované a každá firma musí řešit problém, jak se na tomto médiu prezentovat. V dnešních dnech již dle průzkumů má 95% firem připojení k internetu, tedy možnosti sebe-prezentování jsou jim plně k dispozici. Zatímco obyčejní uživatelé sáhnou vždy při své soukromé prezentaci po co nejlehčím způsobu jak se na internetu prezentovat, většina firem musí vzhledem ke svým potřebám sáhnout po sofistikovanějších řešeních. Firma proto může v zásadě zvolit mezi dvěmi řešeními. Použít redakční systém, či zvolit software vyrobený přímo dle jejích potřeb.

Prezentace pomocí již existujících redakčních systémů způsobuje jistou jednotvárnost v jednotlivých prezentacích a také snadné šíření chyb. Tvorba vlastních aplikací proti tomu znamená unikátní prezentaci, to z toho důvodu že kód je přímo tvořen na navržený design, a také většinou vyšší záruky bezpečnosti z důvodů odlišnosti zdrojových kódů. Z těchto důvodů by se měly firmy většinou rozhodnout pro řešení aplikaci své prezentace aplikaci vytvořenou úplně od základů.

Každá firma si také zakládá na tom aby jejich prezentace byla ta nejlepší vůči jejich konkurenci. Zde je třeba mít stále na zřeteli, že kvalitu prezentace ovlivňuje

nejen vzhled, ale i jak vypadá její pozadí, tedy její zdrojový kód. Protože kvalita kódu ovlivňuje nejen rychlost prezentace a její velikost kterou musí cílová osoba stáhnou, ale i možnost jak se budou provádět její aktualizace, tedy usnadnění administrace a přidávání nových funkcí na které nebylo myšleno v počátcích prezentace.

Rozhodování firem je také o tom zda se práce zadá vlastnímu IT (informační technologie) týmu, či se práce zadá kontraktorům, tedy firmám které které se přímo specializují na vývoj WWW prezentací. U těchto možností může vzniknout mnoho problémů, protože firma musí vyřešit zda bude platit externí firmě za podporu dané aplikace, nebo zaplatí pouze vyškolení vlastních lidi a tedy jestli by nebylo levnější zda by se celá aplikace neměla vytvořit firmou. Zde musí firma přihlédnout jak zkušené lidi v IT má a jak kvalitní by byl výsledný kód.

Právě o kvalitě a obtížnosti práce s jednotlivými programovacími jazyky pojednává předložená bakalářská práce.

1 Cíl práce a metodika

Cílem bakalářské práce je popsat a zhodnotit vlastnosti jednotlivých skriptovacích a programovacích jazyků. Vypsat jejich silné i slabé stránky a pokusit se rozhodnout zda se hodí pro firemní prezentaci na WWW.

Programovací a skriptovací jazyky byly vybrány tak aby bylo prezentováno co nejlépe spektrum dnes používaných jazyků pro tvorbu WWW prezentací. Vybrané jazyky byly poté omezeny pouze na ty co jsou snadno provozovatelné na různých OS a jsou otevřeny a zdarma k použití. Veškeré testování bude probíhat na OS Gentoo Linux.

K porovnání jazyků slouží jak výčet a srovnání jejich funkcí, tak v kapitole Porovnání programovacích a skriptovacích jazyků srovnání účinnosti jednoduchých testovacích skriptů. Skripty byly zvoleny co nejjednodušeji a prezentují často užitou funkci. Načtení dat z výstupu od uživatele a jejich validaci a kontrolu proti podezřelým a špatným vstupům. Vzhledem ke složitosti nastavení a zjednodušení měření byla zvolena varianta kdy je vynechán webový server a používá se přímé zadávání vstupů z konzole.

Porovnává se čas potřebný k vytvoření kódu pokročilým programátorem (autor bakalářské práce) a programátorem-začátečníkem (slečna Anna Janáčková), tak doba trvání tvorby kódu mezi jednotlivými jazyky. Srovnání času psaní kódu mezi jednotlivými jazyky nelze přikládat velký význam, protože při několikátém opakování bude napsaní stejného, či podobného kódu trvat méně času, i když je třeba dodržovat syntaxi jiného jazyka. Další subjektivní charakteristikou je čitelnost a srozumitelnost daného kódu a možnosti jeho údržby.

Popsána budou dále pracovní prostředí, kde aplikace budou pravděpodobně pracovat při svém použití a vývojová prostředí, ve kterých lze dané programy psát. Tento popis bude sloužit jako informativní a při zohlednění v porovnávání naň nebude brán zřetel.

Práce popisuje stav k prvnímu čtvrtletí roku 2008 a veškeré změny v jazycích po tomto datu nezohledňuje.

2 Programovací jazyky pro WWW

Programovací jazyk je nástroj pro přepis algoritmů pro PC. Tento přepis algoritmu se nazývá program. Je určen pro převod konstrukcí od programátora do strojového kódu pochopitelného samotným počítačem.

Programovací jazyky se rozdělují dle několika kritérií do několika kategorií.
Dle postupu při překladu:

- Kompilované programovací jazyky: zdrojový kód se převádí přímo do strojového kódu (C, C++).
- Interpretované programovací jazyky (Java, Perl, PHP, Python, Ruby). V zásadě existují tři možnosti interpretace:
    1. Nepřevádí se přímo do strojového kódu, ale zdrojový kód se vždy spouští přes interpretr, který ho zpracovává (BASH).
    2. Převádí se do tzv. bytecode (mezikódu) a ne do strojového kódu (Python).
    3. Převádí se do strojového kódu po spuštění programu (Java[1]).

[1]Z tohoto důvodu není Java umístěna mezi skriptovací jazyky.

Dle úrovně abstrakčního rozhraní:

- Nižší programovací jazyky (Assembler a jemu podobné).
- Vyšší programovací jazyky (většina dnes používaných jazyků).

2.1 C, C++

C je univerzální procedurální, imperativní programovací jazyk pro PC (osobní počítač) vyvinutý pro užití s OS (operační systém) Unix. Jedná se o nízkoúrovňový jazyk, ve kterém je díky své univerzálnosti napsán interpretr většiny dalších vysokoúrovňových jazyků, kterým se věnují další kapitoly.

Z jazyka C vychází většina dnešních programovacích jazyků, ať už se jedná o PHP či o Javu. Lze to snadno dokázat na ukázkových příkladech, které budou obsahovat kód algoritmu **„Ahoj světe!“**.

Jazyk C++:

Zdrojový kód 2.1: Program „Hello World!“ v C++

```
#include <iostream>
using namespace std;
int main(int argc, char ** argv) {
        cout << "Hello World!" << endl;
        return 0;
}
```

Jazyk C:

Zdrojový kód 2.2: Program „Hello World!“ v C

```
#include <stdio.h>
int main(int argc, char ** argv) {
        printf ("Hello World!\n");
        return 0;
}
```

Samozřejmě zdrojový kód může vypadat i velice chaoticky:

Zdrojový kód 2.3: Zdrojový kód v C

```
#include <stdio.h>
main(t,_,a)
char *a;
{return!0<t?t<3?main(-79,-13,a+main(-87,1-_,
main(-86, 0, a+1 )+a)):1,t<_?main(t+1, _, a ):3,main ( -94, -27+t, a
)&&t == 2 ?_<13 ?main ( 2, _+1, "%s %d %d\n" ):9:16:t<0?t<-72?main(_,
t,"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+,/w{%+,/w#q#n+,/#{l,+,/n{n+\
,/+#n+,/#;#q#n+,/+k#;*+,/'r : 'd*'3,}{w+K w'K:'+}e#';dq#'l q#'+d'K#!/\
+k#;q#'r}eKK#}w'r}eKK{nl]'/#;#q#n')()#}w'){){nl]'/+#n';d}rw' i;# ){n\
l]!/n{n#'; r{#w'r nc{nl]'/#{l,+'K {rw' iK{;[{nl]'/w#q#\
n'wk nw' iwk{KK{nl]!/w{%'l##w#' i; :{nl]'/*{q#'ld;r'}{nlwb!/*de}'c \
;;{nl'-{}rw]'/+,}##'*}#nc,',#nw]'/+kd'+e}+;\
#'rdq#w! nr'/ ')}+}{rl#'{n' ')#}'+}##(!!/")
:t<-50?_==*a ?putchar(a[31]):main(-65,_,a+1):main((*a == '/')+t,_,a\
+1 ):0<t?main ( 2, 2 , "%s"):*a=='/'||main(0,main(-61,*a, "!ek;dc \
i@bK'(q)-[w]*%n+r3#l,{}:\nuwloca-O;m .vpbks,fxntdCeghiry"),a+1);}
```

Tento kód je naprosto validní a po zkompilování vrací anglický text, který se zabývá tím, co osoba dostávala od své přítelkyně v průběhu dvanácti dní od Vánoc.

2.1.1 Shrnutí fundamentálních vlastností jazyka C

- Deklarace: Každá proměnná, či funkce, musí být vytvořena před svým použitím. Každá funkce/proměnná musí mít také předem definovaný typ. Jedná se o identifikaci návratové hodnoty/obsahu. Např.: `int␣soucet;` vytvoří proměnnou soucet, do které lze uložit pouze celá čísla o velikosti integer.

- Každý jednotlivý blok kódu, ať již se jedná o funkci, či podmínku, musí být uzavřen do složených závorek (`{}`), tímto postupem se inspirovalo mnoho jazyků, výjimkou v jazycích které zde popisujeme je Ruby a Python. Výjimkou z tohoto pravidla je blok, ve kterém se nachází pouze jeden příkaz: `for(int␣i=0;i<lines;i++)␣print␣i;` je to samé jako
  `for(int␣i=0;i<lines;i++)␣{␣print␣i;␣}`.

- Pointery (ukazatele): V podstatě to nejlepší, co nám jazyk C může nabídnout. Jedná se o ukazatel do paměti na jinou proměnnou. Tímto můžeme v programu různě předávat pouze ukazatel a ne celý obsah paměti.

- Pole jsou v jazyce C vetšinou považována za statická, tedy je třeba předem definovat jejich velikost, kterou není možno později přesáhnout.

Výhodou jazyka C je převážně přenositelnost na jakoukoliv platformu. Lze jej „vecpat“ do všech možných zařízení od sálových počítačů po mikrovlnku. Vůči ostatním jazykům má nízkou spotřebu zdrojů. Nevýhodou jazyka C pro nasazení ve vývoji webových aplikací je bohužel neexistence jednoduchého interpretru, pro webový server.

Díky tomuto vychází jazyk C nejlépe vzhledem k nárokům na HW (hardware). Oproti tomu má nejvyšší nároky na odbornost programátora, vzhledem k ostatním

jazykům zahrnutým v bakalářské práci. Lze jej spouštět pouze převodníkem jako CGI Skript. Tato metoda je nevýhodná v tom, že vytváří nové vlákno, ve kterém spouští vlastní aplikaci, což způsobuje její značné zpomalení spouštění.

2.1.2 Historie

Jazyk C byl vytvořen v Bellových laboratořích AT&T Denisem Ritchie v rozmezí let 1966 až 1973. Jeho pojmenování bylo zvoleno vzhledem k tomu, že mnoho vlastností bylo převzato z jazyka, který se jednoduše jmenoval B. Cílem bylo napsat jazyk pro snadnou a přenositelnou implementaci Unixu. Na tomto přenosu (Unixu již existujícího pro PDP-7) se jako další podíleli Brian Kernighan a Ken Thompson.[2]

Jazyk C vznikl jako systémový jazyk pro systémové programátory. Po řadu let byla faktickou normou jazyka C kniha *Programovací jazyk C* autorů K&R (Kernighan and Ritchie), jednoduše byl tento typ programovaní nazýval „K&R C“. V tomto C bylo možno provádět nejrůznější triky, které ovšem činily zdrojový text velice nečitelným. Nicméně s postupem času se C díky studentům, kteří se v něm učili na vysokých školách, započalo dostávat do praxe i v jiných než systémových oblastech. Programátoři přirozeně požadovali, aby C bylo nejen mocným prostředkem, ale i bezpečným a přehledným jazykem. Vznikl trend vedoucí k normalizaci jazyka. Teprve díky normalizaci se jazyk mohl rozšířit.[1]

ANSI norma (Americký národní standardizační institut) jazyka C jej definuje jako moderní vyšší programovací jazyk všeobecného použití. Shodný standard definuje ISO (Mezinárodní standardizační organizace). Proto se někdy uvádí toto spojení jako ANSI/ISO norma. ANSI/ISO specifikace jazyka C je bezpečnější nežli K&R C.

Překladače, díky normalizaci jazyka, nabízí řada nezávislých producentů programového vybavení, včetně GNU (rekurzivní akronym pro Gnu's Not Unix, znamenající v tomto případě plnohodnotnou alternativu s dostupnými zdrojovými kódy). Podstatné je, že C je dostupné na prakticky všech platformách. Bude-li ANSI/ISO norma C dodržována při tvorbě programu, existuje vysoká pravděpodobnost, že zdrojový kód bude přenositelný na jiné PC (s jiným procesorem i OS) a program bude možno jak přeložit, tak i spojit s knihovnami a spustit.[2]

ANSI/ISO C je navrženo tak, že kodifikuje existující praktiky. Většina zdrojových textů, napsaných před vydáním normy, je novými překladači akceptována. Přesto je ANSI/ISO C témě novým jazykem vzhledem k K&R C :

- Přidává některé vlastnosti (prototypy funkcí,...). Opravuje tím některé nedostatky K&R definice jazyka C.
- Přehodnocuje konfliktní praktiky, jako například rozdílná pravidla pro redeklaraci datových objektů.
- Vyjasňuje dvojznačnosti. Například zákaz překrytí datových objektů zpracovávaných knihovní funkcí.

S postupem doby se nezastavila ani teorie programování. Jestliže C umožňuje psát programy modulárně, pak dnešní trend vyžaduje i podporu objektového přístupu, kterou přináší C++. V roce 1986 publikoval Stroustrup knihu *The C++ Programming Language*. Počátkem 90. let pak byl i tento jazyk normalizován. Jedná se o objektové rozšíření jazyka C. Jazyk C++ je v současné době jeden z nejrozšířenějších programovacích jazyků.

Po standardizaci v roce 1989 se vývoj zaměřil hlavně na jazyk C++, přesto byl počátkem roku 2000 přijat jako standart ISO 9899:1999 (Zkráceně C99) vztahující se k jazyku C. Tato „novelizace“ přinesla mnoho vylepšení. Jen namátkou:

- Možnost deklarovat proměnné kdekoliv v kódu.
- Nové datové typy: 64B integer, logický typ boolean a jiné.
- Pole s nekonstantní velikostí.

2.2 Java

Java je dnes druhým nejrozšířenějším programovacím jazykem světa. Díky své platformní nezávislosti, jednoduchosti a nízkým nárokům na programátora je využívána v mnoha projektech jak v prostředí WWW tak i pro klasické desktopové (běžící přímo na pc) programy.

„Ahoj světe!“ v jazyce Java:

Zdrojový kód 2.4: Program „Hello World!“ v Javě

```
public class hello_world {
        public static void main(String args[]) {
                System.out.println("Hello World!");
        }
}
```

2.2.1 Shrnutí výhod jazyka Java

- **Jednoduchost:** Toto znamená, že Sun vytvořil Javu co nejvíce podobnou jazyku C++ s tím, že odstranil většinu věcí, které byly matoucí, nepoužívané, či špatně pochopitelné. Mezi ně patří zejména přetěžování operátorů a několikanásobnou dědičnost. Největším zjednodušením je však nepoužívání pointerů a užívání garbage collectoru (sběrač odpadu) pro automatickou práci s pamětí.

- **Objektová orientace:** Znamená, že programátor se zaměřuje pouze na data a přístup k těmto datům v aplikaci. Java je velice přísná co se týče objektově orientovaného programováni. Vše musí být naprogramované jako volání metody v objektu (je možno vidět na ukázkovém příkladu 2.4).

- **Distribuovatelnost:** Java je navržena pro podporu síťových aplikací. Tedy obsahuje spoustu interních funkcí pro práci se sítí, od nejjednodušších až po ta nejsložitější volání.

- **Robustnost:** Jazyk obsahuje spoustu způsobů jak včas odhalit chybu v kódu a upozornit na ní programátora. Odstranění pointerů sejmulo z beder programátora starost o paměť a zabránilo nechtěnému přepisování paměti. Práce s výjimkami usnadnila zpracování možných chyb, které mohou při běhu programu nastat. Ačkoliv Java neprovádí kontrolu kvality kódu, velice ji usnadňuje.

- **Bezpečnost:** Jakožto jazyk navržený pro práci v síti implementuje mnoho bezpečnostních mechanismů tak, aby nedošlo ke spuštění podezřelého kódu. Nemá pointery a neumožňuje porušování paměti. Každá třída je uzavřená ve svém jmenném prostoru, ze kterého nemůže nic „rozbít“.

- **Přenositelnost:** Vychází z platformní nezávislosti a programy psané v Javě lze tedy spouštět všude, kde je možno spustit JVM (Java virtuální stroj).
- **Výkon:** Rychlost aplikací je vyšší nežli rychlost aplikací psaných v plně interpretovaných jazycích a téměř dosahuje rychlosti C/C++.

Java je jednoduchý a zároveň mocný jazyk, který byl přímo navržen pro síťové aplikace. Jedinou nevýhodou vůči ostatním jazykům, kterou je nutno zdůraznit, je vysoká náročnost na systémové zdroje. Většinu této náročnosti lze odstranit důrazem na kvalitu a čistotu kódu, ale k uskutečnění takové věci je třeba mít k dispozici zkušeného (a drahého) profesionála. Java je vhodná pro nasazení do tzv. Enterprise řešení, kde nezáleží na vysoké spotřebě zdrojů a je snadné navýšit kapacitu HW.

2.2.2 Historie

Vývoj programovacího jazyka Java započal v roce 1990 vytvořením Green Teamu, vedeným Jamesem Goslingem. Tento tým byl sestaven pro vytvoření stručného a jednoduchého jazyka pro zápis programů na podnět tehdejšího viceprezidenta Sun Microsystems Billa Joye. Cílovým určením pro Javu měla být spotřební elektronika.

2.2.2.1 Oak

Nejprve pro svou práci používal Green Team jazyk C++. Tento jazyk nebyl díky svým rozšířením shledán vhodným. Poté následoval pokus s jazykem Pascal, který byl také shledán nevhodným. Nakonec Green Team návrh jazyk Oak (anglicky dub), podle stromu, který rostl před jejich kancelářemi. Za předchůdce tohoto jazyka lze považovat jazyk C++. Jazyk byl vytvořen jako interpretovaný, aby byl naprosto nezávislý na platformě. V této době byly požadavky Oaku na zdroje velice nízké.[3]

2.2.2.2 Star 7

Po roce vývoje bylo představeno zařízení Star 7. Jednalo se o prototyp zařízení, spojující PDA (osobní digitální asistent) a dálkové ovládání. Byl vybaven procesorem architektury RISC, dotykovým displejem a umožňoval práci s aplikacemi v jazyce Oak. Bohužel neexistovala velká poptávka po zařízení tohoto typu, proto se nikdy nezačalo masově vyrábět.

Při zveřejnění tohoto zařízení Patrik Naughton prohlásil „in 18 months, we did the equivalent of what 75-people organizations at Sun took three years to do – an operating system, a language, a toolkit, an interface, a new hardware platform,... “, což ve volném překladu znamená „za 18 měsíců jsme udělali stejnou práci, která by týmu 75ti lidí ze Sunu trvala tři roky. Vytvořit operační systém, jazyk, toolkit, rozhraní a novou HW platformu,... “.[3]

2.2.2.3 FirstPerson

V listopadu roku 1992 se Green Project přejmenoval na FirstPerson. Díky neúspěchu Javy v oblasti spotřební elektroniky byl Sun na vážkách kam se bude dál s Javou ubírat. V roce 1993 Sun začal jednání s 3DO o dodávání OS do jejich set-top-boxů. Jednání ovšem selhala a dohoda nebyla nikdy uzavřena. Z dnešního pohledu se dá říci, že to bylo štěstí vzhledem k problémům, které později mělo 3DO. Další pokus byl v prosazení Oaku jako prostředí interaktivní TV, který taky selhal počátkem roku 1994.[3]

Vzhledem k neúspěchům Sun většinu zaměstnanců přeřadil do jiných projektů. Několik zaměstnanců zůstalo a pracovalo na prosazení Javy v sítových aplikacích.

2.2.2.4 Java a WWW

V půli roku 1994 byl zahájen projekt „Liveoak“, který byl určen pro práci s aplikacemi na internetu, tento nápad dostal jeden z vývojářů, když programoval webový prohlížeč. Toto byl okamžik kdy se Java dostala ze svých neúspěchů a začala se prosazovat. Později MS (Microsoft) oznámil, že jejich prohlížeč bude Javu podporovat, což potvrdilo roli Javy v prostředí WWW.

Ač dnes neslouží svému původnímu návrhu, spousty věcí zakomponovaných v Javě je využito. Ať se jedná o návrh platformní nezávislosti, či vysokou bezpečnost kódu.

3 Skriptovací jazyky pro WWW

Skriptovací jazyk je takový programovací jazyk u kterého se neprovádí kompilace, ale interpretace zdrojového kódu. Což znamená, že kód se nepřevádí do strojového, ale zůstává ve své původní podobě a do kódu srozumitelného pro počítač je převeden až při svém spuštění pomocí interpretru.

Vhodným využitím skriptovacích jazyků je např. rozšíření libovolného projektu o parametrickou část, kdy odpadá nutnost překompilování programu při změnách v parametrické části. Také je vhodným nástrojem pro tvorbu dynamických internetových stránek. Tento aspekt je dále probírán v následujících kapitolách.

Výhody:

- Není třeba rekompilovat po každé změně ve zdrojových kódu.
- Snadnější údržba kódu.
- Abstraktní pole: prvky v poli nejsou indexovány, ale přistupuje se k nim pomocí klíče.

Nevýhody:

- Programy se převádějí do kódu srozumitelného pro PC až při spuštění programu a rychlost je tedy mnohem nižší nežli programů optimalizovaných a kompilovaných.
- Vyšší paměťová náročnost.

Hlavní a klíčovou vlastností vlastní téměř všem skriptovacím jazykům je netypovost proměnných. Což znamená, že není třeba předem definovat jakého typu bude proměnná.

3.1 Perl

Perl jakožto jazyk, který v sobě přímo implementuje awk, sed a grep (mocné *NIXové utility pro práci s textem), umožňuje dokonalou a snadnou práci s libovolným textem díky regulárním výrazům.

Díky objektové orientaci umožňuje snadnou práci s kódem (ačkoliv umožňuje programátorovi objekty naprosto nevyužít) a rychlé přenesení objektového návrhu do praxe.

Pozoruhodnou vlastností většiny programátorů jazyka Perl je inklinace k psaní kódů jako oneliner, tedy tak, aby se veškerý kód vešel na jednu řádku, nebo se tomuto stavu velice blížil (viz 3.1).

Zdrojový kód 3.1: Řešitel sudoku v jazyce Perl

```
#!/bin/perl
$_=$'.$_.$'.<>;split//;${/[@_[map{$i-($i="@-")%9+$_,9*$_+$i%;
9,9*$_%26+$i-$i%27+$i%9-$i%3}0..8]]/o||do$0}for/0/||print..9;
```

Jako každý skriptovací programovací jazyk má velice jednoduchý výstup na terminál. Program „Ahoj světe!“ je díky tomu velice krátký a jednoduchý:

Zdrojový kód 3.2: Program „Hello World!“ v Perlu

```
#!/usr/bin/perl
say 'Hello World!';
```

3.1.1 Klíčové znaky jazyka Perl

- Perl obsahuje rozhraní pro integraci databází (DBI), které umožňuje snadné propojení se všemi známými databázemi např. Oracle, Postgres, MySQL.
- Perl spolupracuje s HTML (jazyk popisu stránek) i dalšími značkovacími jazyky (XML).
- Perl plně podporuje UNICODE (standard pro kódování znaků), tedy lze do něj psát znaky v jakémkoliv jazyce (včetně Klingonštiny).
- Perl podporuje jak procedurální, tak objektově orientované programování, záleží tedy na programátorovi jaký přístup zvolí pro svou aplikaci.
- Interpretr Perlu je možno zabudovat do mnoha zařízení. Což umožňuje velkou platformní nezávislost.

- Perl je, díky precizním možnostem pro manipulaci s texty, nejpopulárnější programovací jazyk pro WWW a díky CPAN databázi (viz Historie jazyka Perl) umožňuje snadnou a rychlou tvorbu aplikací.

Díky těmto vlastnostem je Perl velice vhodným kandidátem pro nasazení ve všech projektech, které se na WWW vytvářejí. Díky své jednoduchosti a CPAN databázi usnadňuje programátorovi začátky, a poté zrychluje tvorbu aplikací.

3.1.2 Historie

Poprvé představen v zimě roku 1987 Larrym Wallem. Jazyk se stal velice oblíbeným a začal se velice rozšiřovat. S ročním odstupem poté vyšly verze 2.0 a 3.0, které přinesly lepší podporu regulárních výrazů a podporu pro binární datové proudy.

Do roku 1991 byla jedinou existující dokumentací manuálová stránka (`man␣perl`), která byla neustále zvětšována. V roce 1991 byla vydána kniha „Programming in perl“, která se stala ve své podstatě referenční příručkou k jazyku Perl. U příležitosti vydání této knihy bylo navýšeno číslo verze jazyka na 4.0 a logo bylo převzato z obalu knihy (Kráčející velbloud).

V roce 1994 byl vydán Perl 5.0. Interpretr byl s touto verzí přepsán od základu. Jazyk již zvládal objekty, lokální proměnné a moduly. Modularita umožnila podstatné rozšiřování schopností jazyka bez nutnosti úprav interpretru. To usnadnilo stabilizaci jádra interpretru a umožnilo přidávání rozšíření komukoliv.

Velice významným okamžikem pro rozšíření jazyka byl rok 1995, kdy byl vytvořen CPAN (Comprehensive Perl Archive Network) repositář, ve kterém je možno

najít spousty ukázkových zdrojových kódů. Tento repositář usnadňuje a zrychluje vývoj aplikací.

3.1.2.1 Název

Název PERL vychází z původního přání autora na jméno PEARL, což je již jiný programovací jazyk a z toho důvodu bylo jedno písmenko z názvu odstraněno.

Dnes se považuje za nejvhodnější „**P**ractical **E**xtraction and **R**eport **L**anguage“, ačkoliv dle autorů se zkratka PERL nemá rozepisovat a je to opravdu pouze PEARL ochuzené o jedno písmenko. Další z mnoha vtipných rozepsání, které je možno nalézt v emailových konferencích je **P***athologically* **E***clectic* **R***ubbish* **L***ister*.[4]

3.2 PHP

PHP je dnes jedním z nejpoužívanějších programovacích jazyků pro tvorbu dynamických webových stránek. PHP funguje na mnoha platformách, operačních systémech a webových serverech. Je nabízeno zdarma. Je ovšem umožněno přes placený ZEND kompiler převést kód do binární formy, která je rychlejší nežli forma interpretovaná.

Hlavním směrem působení PHP je tvorba WWW, i když umožňuje i tvoření zcela samostatných skriptů. Většina návodů k programování v PHP je na internetu psaná pro začátečníky a základní pochopení jazyka je tudíž velice snadné. Bohužel obliba mezi začátečníky vedla k velkému množství špatně napsaných návodů z nichž některé postupy mohou vést k velkým bezpečnostním rizikům.

V půli roku 2007 bylo PHP užito na více nežli 20 milionech WWW stránek a to z něj dělá velice častý cíl crackerů, kteří se snaží do aplikací „nabourat“ pomocí ať už chyb v PHP, či pomocí XSS (umožnění spouštění kódu jiné stránky). Díky těmto nevýhodám není dnes vhodné příliš používat PHP pro kritické nasazení, pokud programátoři píšící aplikaci nejsou již zkušení a dokáží většině problémů zabránit.

Program „Ahoj světe!“ v jazyce PHP:

Zdrojový kód 3.3: Program Hello World! v PHP

```
<?php
        echo "Hello World!";
?>
```

3.2.1 Historie

PHP/FI (Personal Home Page/Form Interpreter) začínal jako soubor CGI skriptů v jazyce Perl, později přepsaných do jazyka C, napsaných Rasmusem Lerdofem v roce 1995. Původním cílem bylo pouze usnadnit tomuto programátoru jednoduchý přístup k různým statistikám jeho webových stránek.

Jazyk již tou dobou obsahoval věci, které se používají ještě dnes. Např. zápis proměnných stejně jako v jazyce Perl, či zabudovanou podporu pro HTML syntaxi. V roce 1997 vyšlo PHP/FI 2.0 a došlo k přepsání jádra do jazyka C. Touto dobou se jazyk používal již na 1% WWW. Ačkoliv mnoho lidí přispívalo do projektu stále to byl projekt, který kontroloval pouze Ramus Lerdof.[9]

V roce 1998 vychází PHP 3.0, které se již v mnoha ohledech podobá dnešnímu PHP. Bylo napsáno Andi Gutmansem a Zeev Suraskem jako kompletní přepsání původního PHP/FI. Došlo k přejmenování programovacího jazyka, kde PHP znamená nyní rekurzivní akronym „PHP: Hypertext Preprocessor“. Ačkoliv PHP bylo značně populární stále neobsahovalo objektový model a dalo se psát nejvýše procedurálně.[9]

Krátce po oznámení PHP 3.0 byly zahájeny práce na PHP 4.0, které si kladlo za cíl zvýšení výkonu jazyka, modularity a objektového modelu. Tato verze v roce 1999 přešla pod nově stvořený *Zend Engine* a v roce 2000 vyšla jako pokračovatel PHP 3.0. Tato verze již obsahovala snadnou prci se sezeními (sessions), podporu pro různé web-servery a spousty bezpečnostních vylepšení. Hlavní novinkou u této verze bylo využívání lokálních proměnných.[9]

Od roku 2004 je k dispozici PHP 5.0, které vylepšilo objektový model do použitelné roviny a opravuje spoustu bezpečnostních chyb. Tato verze je řízena Zend Enginem verze 2.0.

3.3 Python

Python byl vyvinut jako jazyk pro neprogramátory, tedy jeho syntaxe je velice jednoduchá a snadno pochopitelná. Přebírá spoustu vlastností jak s funkcionálního programování, tak plně implementuje objektový model.

Díky své jednoduchosti umožňuje tento jazyk aby se programátor naučil základní struktury a postupy velice snadno. Python se tedy dá naučit i do dvou dnů.

Ukázka programu „Ahoj světe!“:

Zdrojový kód 3.4: Program „Hello World!“ v jazyce Python

```
#!/usr/bin/python
print "Hello World!"
```

Zápis veškerého kódu je rozdílný vůči všem ostatním programovacím jazykům. Řádky nejsou zakončovány středníkem a bloky kódu se neuzavírají do složených závorek. Příslušnost do bloku kódu se rozlišuje tabulátorem.

Zdrojový kód 3.5: Ukázka různých bloků kódu v Pythonu

```
#!/usr/bin/python

def zmenVelikost (nazevSouboru, rozliseni):
        dirname = "zmenseno"
        try:
                os.mkdir(dirname)
        except:
                pass
        novyNazev=os.path.join(dirname,nazevSouboru)
        try:
                os.path.getsize(novyNazev)
        except:
                print "Prevadim %s na %s" %(nazevSouboru, novyNazev)
```

Jazyk Python obsahuje díky své modularitě spoustu rozšíření, které umožňují naprogramovat vše velice jednoduše. Díky jednotnému zápisů programů zdrojový kód zůstává velice podobný a je snadno udržovatelný jinými programátory. Je to ideální jazyk jak pro začátečníky, tak pro programátory, kteří se chtějí později přesunout k funkcionálnímu programování. Vysoká bezpečnost výstupního kódu umožňuje využití tohoto jazyka pro všechny projekty bez rozdílu.

3.3.1 Historie

Práce na Pythonu započala v roce 1989, kdy tehdejší zaměstnanec CWI Guido Van Rossum začal pracovat na distribuovaném OS Amoeba, který vychází z programovacího jazyka ABC. Jazyk ABC byl vytvořen v 80. letech jako programovací jazyk pro neprogramátory, z důvodů naprosté nevhodnosti BASICU a standardních programovacích jazyků pro tyto účely.[8]

V roce 1991 byla vydána první veřejná verze Pythonu s číslem 0.9, která obsahovala třídy, práci s výjimkami či základní datové typy. Na tuto verzi navázala v roce 1994 verze 1.0, ve které byly přidány funkce z funkcionálního programování inspirované jazykem Lisp.

Python 2.0, který vyšel v roce 2000, obsahoval další klíčové funkcionální vlastnosti převzaté z jazyka SETL a Haskell. Konstrukce těchto převzatých funkcí je velice podobná konstrukci z těchto jazyků. Dále obsahoval garbage collector (systém pro automatickou práci s pamětí).

Nyní se vyvíjí Python 3.0, jinak také známý jako Python 3000, který odstraňuje chyby v návrhu původního jazyka Python, díky čemuž ztrácí zpětnou kompatibilitu. Hlavním principem všech změn v této verzi je odstranění redundance vlastností tak, aby zůstala pouze jediná cesta tvorby daného algoritmu.

3.4 Ruby

Ruby je moderní programovací jazyk plně implementující objektový model a snažící se o co největši jednoduchost v nárocích kladených na programátora. V posledních letech oslabuje pozici PHP jako hlavního jazyka pro tvorbu dynamických webových stránek.

Díky svému integrovanému frameworku RoR (Ruby on Rails) a repositáři Gemů (modulární rozšíření) je tvorba programů v Ruby velice jednoduchá a rychlá. Vytvoření blogovacího rozhraní autoru této bakalářské práce trvalo přibližně 3 hodiny čistého času bez předchozích zkušeností s tímto jazykem.

Jazyk je rozumně bezpečný a snaží se programátora učit používat správné postupy při zápisu kódu. Bohužel nepodporuje UNICODE a tedy je opravdu nevhodný pro tvorbu projektu s více jazykovými mutacemi.[5]

„Ahoj světe!“ v jazyce Ruby:

Zdrojový kód 3.6: Program „Hello World!“ v jazyce Ruby

```
#!/usr/bin/ruby
puts "Hello World!"
```

Jazyk Ruby je vhodný pro programátory přecházející z jazyka PHP, kteří se nechtějí učit příliš nových věcí. Přináší silný typový model a plnou podporu objektů. Bohužel kvůli nepodpoře UNICODE se nemůže uvažovat o jeho nasazení na velké projekty s mezinárodním využitím.

3.4.1 Historie

Jazyk byl vytvořen panem Yukihiro Matsumoto. Práce na něm započaly v roce 1993 a po dvou letech byla zveřejněna první verze. Ruby byl záměrně pojmenován jako drahokam k protikladu vůči programovacímu jazyku Perl.[5]

Dnes je ve vývoji verze 1.9, která přináší podporu kompilace kódu do bitecode, což výrazně urychlí spouštění aplikací.

4 Pracovní a vývojová prostředí

Pod pojmem pracovní prostředí je pro potřeby bakalářské práce považováno prostředí běhu daných skriptů, či programů. Pro WWW prezentace je nejčastějším prostředím webový server a tedy bude následovat jejich stručný popis. V popisu jsou zohledněny pouze open source (programy s otevřeným, volně dostupným zdrojovým kódem) varianty, z důvodů nižších nákladů na jejich pořízení a provoz.

Pod pojmem vývojového prostředí se myslí prostředek pro zápis kódu. Od komplexního IDE (integrované vývojové prostředí) po jednoduchý editor. Pro potřeby bakalářské práce budou popsány jednoduchá prostředí, která jsou dostupná pod OS Gentoo Linux.

4.1 Web-servery

Webový server je počítačový program, který se stará o zpracování HTTP (hypertext) dotazů od webových prohlížečů, či jiných klientů a předává zpracovaný výstup zpět.

4.1.1 Apache

Apache byl první webový server, který bylo možno snadno nastavovat a používat. Umožňuje správu virtuálních hostů, tedy že na jedné IP adrese (číselné označení pc) je možno provozovat více domén (např. www.seznam.cz a www.centrum.cz lze mít na 1 PC).

Dále umožňuje přes modulární rozšíření pro integraci jazykových interpretrů jejich včlenění přímo do webového serveru. Tato vlastnost umožňuje výrazné zrychlení spouštění skriptů. Modulární rozšíření nejsou součástí serveru implicitně, ale je možno jejich stažení z WWW. Moderní linuxové distribuce se při instalaci zeptají administrátora zda a jaká rozšíření si přeje instalovat.[7]

Tento server je považován za jeden z hlavních důvodů počátku rozšiřování OS založených na jádru Linux, ale dnes je již portován (tzn. lze jej spouštět) i na dalších platformách včetně MS Windows.

Největší výhodou Apache je interakce s dynamickými prostředky a nevýhodou přetěžování při vysokém množství dotazů (1000 dotazů/s) na statický obsah. Nejvýhodnější možností se jeví kombinace zde zmíněných serverů, kde Lighttpd slouží jako přístup ke statickému obsahu a Apache se stará o dynamiku.

4.1.2 Lighttpd

Lighttpd je odlehčený webový server se snahou o co nejmenší zátěž systému. Cílem projektu je vytvořit provozuschopný server na velice zatížené webové prezentace, či server na zpracování statického obsahu. Na serveru je možno spouštět

všechny zde popsané jazyky jako CGI skripty a pouze PHP přes speciální rozhraní s podobným výkonem jako na serveru Apache.

Lighttpd se hodí pro provoz na velice zatížené prezentace, ale jeho absence podpory skriptů a jazyků z něj dělá nevhodné řešení pro samostatné použití. V kombinaci se serverem Apache je to však vhodná volba.[6]

4.2 Vývojová prostředí

Vývojové prostředí je software, který se snaží usnadnit programátorům vývoj a tvorbu aplikace. Většinou obsahuje editor a dále i např. kompilátor či debugger (prohlížeč zdrojového kódu za běhu pro snadné odhalování chyb).

4.2.1 Vim

Vim je textový editor vycházející z populárního editoru VI. Obsahuje 3 režimy běhu: příkazový, vkládací a režim příkazové řádky. Tento editor je na první pohled velice nepřátelský, na druhou stranu má velice promyšlené rozšíření a možnosti.

- **Příkazový režim:** v tomto režimu se nachází editor po svém spuštění. Očekává příkazy ve formě jednoklávesových zkratek.
- **Vkládací režim:** v tomto režimu bude vše chápáno jako vkládaný text.
- **Režim příkazové řádky:** tento režim je prezentován dvojtečkou v levém dolním rohu. Jedná se o jednorázový režim, kdy se editor po zadání příkazu neprodleně vrací do předchozího režimu.

Editor Vim umožňuje práci stejně pohodlně jako běžná IDE, např. zvládá doplňování kódu i kontrolu pravopisu. Po nastavení tohoto editoru je rychlost práce vyšší, než v grafických editorech vzhledem k nepotřebě užívání myši (viz příloha A.1).

Vybrané vlastnosti editoru Vim:

- Logické ovládání
- Podpora regulárních výrazů
- Podpora obnovy textu
- Funkce undo
- Zvýrazňování syntaxe
- Sloupcové bloky
- Doplňování příkazů
- Kontrola párových závorek
- Makra

4.2.2 Kate

Editor Kate je implicitní rozšířený editor pro desktopové prostředí KDE a využívá vysoké provázanosti se systémem. Zvládá všechny funkce klasických IDE, včetně těch zmíněných u editoru ViM.

Díky grafickému rozhraní (viz příloha A.2), je editor uživatelsky přijatelnější než Vim a lze jej využít pro snadnou tvorbu jak velkých projektů, tak jednoduchých skriptů.

5 Porovnání programovacích a skriptovacích jazyků

Porovnáním jazyků je myšlena doba potřebná pro vytvoření jednotlivých algoritmů a čas potřebný k jejich úspěšnému proběhnutí..

Za algoritmus k implementaci bylo vzato validování vstupních dat. Validují se předem definovaná data a je tedy pouze třeba implementovat algoritmus a poté ho převést pod všechny zvolené jazyky.

Zdrojový kód 5.1: Očekávaný výstup k validovaným datům - odchylky způsobují regulární výrazy

```
u       Hodnota: http://www.gmail.com/ je validni!
u       Hodnota: htk://wwa.se;znam.cz/ neni validni!
s       Hodnota: Toto je prosty text. je validni!
e       Hodnota: pepa.jozef@hornidolni.ltd je validni!
e       Hodnota: pepa.karel@dolni@horni.net neni validni!
```

Z důvodů omezení bylo limitem pro vytvoření těchto skriptů hranice 3 hodin. Tímto sítem neprošel programovací jazyk C z důvodů vysoké složitosti. Jazyk C umožňuje napsání této jednoduché aplikace s využitím knihovny `regex.h` z balíku

GNU GCC, která implementuje regulární výrazy přesně dle POSIX. Pokud by byla aplikace napsána v C byla by s nejvyšší pravděpodobností nejvýkonnější jako standalone (samostatná) aplikace, ale při spuštění přes webový server by měla drobné ztráty z důvodů spouštění jako CGI skript.

Další skript který nebyl napsán je Perl ze strany začátečníka. Taktéž pro přesažení limitu tří hodin.

Z výše zmíněných důvodů jsou tyto dva jazyky vynechány ve srovnání.

5.1 Java

Zdrojový kód 5.2: Výstup skriptu začátečníka v jazyce Java

```
* Toto je prosty text. => spravne
* pepa.jozef@hornidolni.ltd => spravne
* pepa.karel@dolni@horni.net => spatne
* htk://wwa.se;znam.cz/ => spatne
* http://www.gmail.com/ => spravne
```

Zdrojový kód 5.3: Výstup skriptu pokročilého programátora v jazyce Java

```
s       Hodnota Toto je prosty text. je validni!
e       Hodnota pepa.jozef@hornidolni.ltd je validni!
e       Hodnota pepa.karel@dolni@horni.net neni validni!
u       Hodnota http://www.gmail.com/ je validni!
u       Hodnota htk://wwa.se;znam.cz/ je validni!
```

Doba tvorby pogramu je výrazně vyšší z důvodu nepřílišné znalosti jazyka java pokročilým programátorem. Tvorba tedy trvala 1 hodinu a doba běhu je také vyšší nežli u ostatních jazyků 0,147 s a zdrojový kód lze nalézt v příloze A.1. Pro začátečníka byla doba tvorby 1 hodinu 50 minut a doba běhu 0.138 s a zdrojový kód lze nalézt v příloze A.2.

5.2 PHP

Zdrojový kód 5.4: Výstup skriptu začátečníka v jazyce PHP

```
text:
        * Toto je prosty text. valid
email:
        * pepa.jozef@hornidolni.ltd valid
        * pepa.karel@dolni@horni.net valid
url:
        * http://www.gmail.com/ invalid
        * htk://wwa.se;znam.cz/ invalid
```

Zdrojový kód 5.5: Výstup skriptu pokročilého programátora v jazyce PHP

```
s       Hodnota: "Toto je prosty text." je validni.
e       Hodnota: "pepa.jozef@hornidolni.ltd" je validni.
e       Hodnota: "pepa.karel@dolni@horni.net" je validni.
u       Hodnota: "http://www.gmail.com/" je validni.
u       Hodnota: "htk://se;znam.cz/" neni validni!
```

Doba trvání tvorby programu byla pro začátečníka 1 hodina a pro pokročilého 32 minut. Jak je možno vidět u zdojových kódů v příloze A.3 a A.4. Oba algoritmy

jsou si podobné a tedy i jejich doba trvání je podobná, pro začátečníka 0,041s a pro pokročilého 0,037s. Jediný problém je s výstupem validací u začátečníka, což může být odstraňeno nahrazením regulerních výrazů.

5.3 Python

Zdrojový kód 5.6: Výstup skriptu začátečníka v jazyce Python

```
url:
        * http://gmail.com/      [ invalid ]
        * htk://wwa.se;znam.cz/ [ invalid ]
text:
        * Toto je prosty text.   [ invalid ]
email:
        * pepa.jozef@hornidolni.ltd      [ invalid ]
        * pepa.karel@dolni@horni.net     [ invalid ]
```

Zdrojový kód 5.7: Výstup skriptu pokročilého programátora v jazyce Python

```
u       Hodnota: http://www.gmail.com/ je validni!
u       Hodnota: htk://wwa.se;znam.cz/ neni validni!
s       Hodnota: Toto je prosty text. je validni!
e       Hodnota: pepa.jozef@hornidolni.ltd je validni!
e       Hodnota: pepa.karel@dolni@horni.net neni validni!
```

Validace v Pythonu vrací správné hodnoty u pokročilého programátora, ovšem u začátečníka vše označí za chybné hodnoty, lze řešit nahrazením regulárních výrazů. Doba tvorby těchto skriptů je podobná jak pro začátečníka 2 hodiny, tak pro pokročilého 42 minut. Doba běhu skriptu je taktéž podobná 0.048 s pro začátečníka

a 0.050 s pro pokročilého. Zdrojové kódy zle nalézt pro pokročilého v příloze A.5 a pro začátečníka v příloze A.6.

5.4 Ruby

Zdrojový kód 5.8: Výstup skriptu začátečníka v jazyce Ruby

```
text:
        * Toto je prosty text. => spravne
url:
        * http://www.gmail.com/ => spatne
        * htk://wwa.se;znam.cz/ => spatne
email:
        * pepa.jozef@hornidolni.ltd => spravne
        * pepa.karel@dolni@horni.net => spravne
```

Zdrojový kód 5.9: Výstup skriptu pokročilého programátora v jazyce Ruby

```
e       Hodnota: pepa.jozef@hornidolni.ltd neni validni!
e       Hodnota: pepa.karel@dolni@horni.net neni validni!
s       Hodnota: Toto je prosty text. je validni!
u       Hodnota: http://www.gmail.com/ neni validni!
u       Hodnota: htk://wwa.se;znam.cz/ neni validni!
```

Zdrojový kód je možno nalézt v přílohách A.7 a A.8 a ani jeden kód nevrátil správný výsledek. Toho by bylo možno dosáhnout úpravou regulárních výrazů. Ruby má nejvyšší rychlost zpracování regulerních výrazů, kolem 0.015s což je 4x rychlejší nežli jazyk PHP a 5x rychlejsi nežli jazyk Python. Vychází tedy nejlépe co do času běhu programu.

5.5 Tabulka časů

Jazyk	**Časy začátečníka**		**Časy pokročilého**	
	Doba tvorby	Doba běhu	Doba tvorby	Doba běhu
Java	1 h 50 min	0.138 s	1 hod	0.147 s
PHP	1 h	0.041 s	32 min	0.037 s
Python	2 h	0.048 s	19 min	0.050 s
Ruby	1,5 h	0.015 s	30 min	0.010 s

Tabulka 5.1: Časy potřebné pro tvorbu a časy běhu jednotlivých algoritmů ve vybraných jazycích

6 Závěr

Závěrem práce lze konstatovat, že všechny výše zmíněné programovací jazyky jsou si velice podobné a tudíž kvalitní. Umožňují programátorům velikou volnost a snaží se usnadňovat odhalování chyb. Díky modulárnosti jazyků je snadné přidávat a odebírat potřebné komponenty, či vytvořit komponentu vlastní, což usnadňuje tvorbu dalších projektů.

Vytvořit WWW prezentaci, která bude kvalitní, umožňují všechny jazyky a lze očekávat, že s rozvojem jazyků se rychlost a kvalita bude nadále zvyšovat. Z těchto důvodů se vyplatí vytvářet přehledné a snadno udržovatelné aplikace, které bude možné snadno rozšířit o nové vlastnosti bez nutnosti úplného přepsání.

Testované jazyky lze rozdělit dle jejich kvalit na dvě kategorie. Kategorii vhodnou pro nasazení na velké firemní prezentace, kde je třeba zohlednit bezpečnostní rizika a dodržení vysoké kvality výstupního kódu. A kategorii vhodnou pro osobní prezentace, kde příliš nezáleží na zabezpečení a kvalitě kódu na výstupu, a není tudíž třeba složitých a riskantních konstrukcí, které jsou v těchto jazycích obtížně uskutečnitelné.

V současné době je v České republice podporován hromadně pouze jazyk PHP a jen v některých případech jazyk Ruby. Je tedy nutno pro nasazení jiného jazyka vytvoření vlastního serveru, či zaplacení podpory, které bude zajisté dražší nežli využití jazyků, které jsou již podporovány. Jistě se tato situace zlepší pokud více velkých zákazníků bude požadovat podporu i pro jiné a kvalitnější jazyky.

Klíčovým bodem této práce je předpoklad kvalitních programátorů a jejich chuť učit se novým věcem. Pokud programátor není dobrý, výsledný kód může pracovat dle požadavků, ale jeho údržba může být v konečném důsledku mnohem dražší, nežli zaplacení profesionálů k prvotnímu vytvoření dané aplikace. Pokud se programátoři naučí s libovolným programovacím jazykem, který je zde zmíněny, budou schopni své znalosti využívat i v jiných oblastech IT, ve kterých firma operuje.

Seznam zdrojových kódů

2.1 Program „Hello World!“ v C++ 10
2.2 Program „Hello World!“ v C 11
2.3 Zdrojový kód v C 11
2.4 Program „Hello World!“ v Javě 15
3.1 Řešitel sudoku v jazyce Perl 22
3.2 Program „Hello World!“ v Perlu 22
3.3 Program Hello World! v PHP 25
3.4 Program „Hello World!“ v jazyce Python 27
3.5 Ukázka různých bloků kódu v Pythonu 27
3.6 Program „Hello World!“ v jazyce Ruby 29
5.1 Očekávaný výstup k validovaným datům - odchylky způsobují regulární výrazy 36
5.2 Výstup skriptu začátečníka v jazyce Java 37
5.3 Výstup skriptu pokročilého programátora v jazyce Java 37
5.4 Výstup skriptu začátečníka v jazyce PHP 38
5.5 Výstup skriptu pokročilého programátora v jazyce PHP 38
5.6 Výstup skriptu začátečníka v jazyce Python 39
5.7 Výstup skriptu pokročilého programátora v jazyce Python 39

5.8 Výstup skriptu začátečníka v jazyce Ruby 40
5.9 Výstup skriptu pokročilého programátora v jazyce Ruby 40
A.1 Zdrojový kód v jazyce Java psaný zkušeným programátorem VI
A.2 Zdrojový kód v jazyce Java psaný začátečníkem VIII
A.3 Zdrojový kód v jazyce PHP psaný zkušeným programátorem X
A.4 Zdrojový kód v jazyce PHP psaný začátečníkem XI
A.5 Zdrojový kód v jazyce Python psaný zkušeným programátorem . . . XIII
A.6 Zdrojový kód v jazyce Python psaný začátečníkem XIV
A.7 Zdrojový kód v jazyce Ruby psaný zkušeným programátorem XVI
A.8 Zdrojový kód v jazyce Ruby psaný začátečníkem XVI

Seznam obrázků

A.1 Ukázka zdrojového kódu bakalářské práce v editoru ViM XIX
A.2 Ukázka zdrojového kódu bakalářské práce v editoru Kate XX

Literatura

[1] **C (programming language)**, [online] [cit. 2008-04-13]
<http://en.wikipedia.org/wiki/C_(programming_language)>

[2] Saloun Petr, **Programování v jazyce C**, [online] [cit. 1996-12-03]
<http://docs.linux.cz/programming/c/c_saloun/kap01.htm>

[3] Marc Abrams, **World Wide Web - Beyond the Basics**, Prentice Hall. 1998.

[4] **The Timeline of Perl and its Culture**, [online] [cit. 2001?]
<http://history.perl.org/PerlTimeline.html>

[5] **Ruby (programming language)**, [online] [cit. 2008-03-18]
<http://en.wikipedia.org/wiki/Ruby_(programming_language)>

[6] **Lighttpd**, [online] [cit. 2008-03-21]
<http://en.wikipedia.org/wiki/Lighttpd>

[7] **Apache HTTP Server**, [online] [cit. 2008-03-21]
<http://en.wikipedia.org/wiki/Apache_HTTP_Server>

[8] **Dive in Python**, [online] [cit. 2008-03-18]
<http://www.diveintopython.org/>

[9] The PHP Group, **History of PHP and related projects**, [online] [cit. 2008-03-17]
<http://php.net/history>

A Přílohy

Zdrojový kód A.1: Zdrojový kód v jazyce Java psaný zkušeným programátorem

```
import java.util.regex.*;

public class tomas_validace {
        private static boolean testInput(String sType, String sInput) {
                Pattern re_email = Pattern.compile("^[_A-Za-z0-9-]+(\\.[_A-Za-z0-9-]+)*@[A-Za-z0-9-]+(\\.[A-Za-
                        z0-9-]+)*(\\.[_A-Za-z0-9-]+)");
                Pattern re_url = Pattern.compile("(([a-zA-Z][0-9a-zA-Z+\\-\\.]*:)?/{0,2}[0-9a-zA-Z;/?:@&=+$\\.\\-_
                        !~*'()%]+)?(#[0-9a-zA-Z;/?:@&=+$\\.\\-_!~*'()%]+)?");
                boolean navrat;
                navrat = false;
                sInput = sInput.trim();
                if (sInput != null) {
                        if (sType=="s") {
                                navrat = true;
                        }
                        if (sType=="e") {
                                Matcher m = re_email.matcher(sInput);
                                boolean shoda = m.matches();
                                if (shoda)
                                        navrat = true;
                        }
                        if (sType=="u") {
                                Matcher m = re_url.matcher(sInput);
                                boolean shoda = m.matches();
                                if (shoda)
                                        navrat = true;
                        }
                }
                return navrat;
        }
        private static void validate(String[][] data) {
                int x,y;
                x = data.length;
                for (int i=0; i<x; i++) {
                        String typ;
```

```
                        typ=data[i][0];
                        y = data[i].length;
                        for (int j=1; j<y; j++) {
                                boolean vysledek;
                                String status;
                                String value;
                                value=data[i][j];
                                if (value != null) {
                                        vysledek = testInput(typ,value);
                                        if (vysledek)
                                                status = " je validni!";
                                        else
                                                status = " neni validni!";
                                        System.out.println(typ + "\tHodnota "+ value + status);
                                }
                        }
                }
        }
        private static String[][] readInput(String[][] data) {
                data[0][0]="s";data[1][0]="e";data[2][0]="u";
                //validni
                data[0][1]="Toto je prosty text.";
                data[1][1]="pepa.jozef@hornidolni.ltd";
                data[2][1]="http://www.gmail.com/";
                //nevalidni
                data[1][2]="pepa.karel@dolni@horni.net";
                data[2][2]="htk://wwa.se;znam.cz/";
                return data;
        }
        public static void main(String[] args) {
                String[][] data = new String[3][3];
                data = readInput(data);
                validate(data);
        };
}
```

Zdrojový kód A.2: Zdrojový kód v jazyce Java psaný začátečníkem

```
/* oink */
import java.util.regex.*;

class Pole {
        String retezec;
        String typ;

        Pole(String retezec, String typ){
                this.retezec = retezec;
                this.typ = typ;
        }

        public String typ() { return this.typ; }
        public String retezec() { return this.retezec; }
}

class Formular {
        Pole [] arr;
        int lastIndex = 0;
        String text = "<(\\/|\\?)*[A-Za-z0-9 ]*(\\/|\\?)*>";
        String email = "[A-Za-z0-9\\-_\\.]+\\@[A-Za-z0-9\\-_]+\\.[a-zA-Z]+$";
        String url = "(http://)?(www\\.)?[A-Za-z]+\\.[a-z]+/?$";

        Formular(int pocet_poli) {
                this.arr = new Pole[pocet_poli];
        }

        public void add(String prvek, String typ) {
                Pole oink = new Pole(prvek, typ);
                this.arr[this.lastIndex] = oink;
                this.lastIndex++;
        }

        public void validovat() {
                for (int i = 0; i < this.arr.length; i++) {
                        System.out.print("* "+this.arr[i].retezec()+" => ");
                        String type = this.arr[i].typ();
                        if (type == "t") {
                                if ( Pattern.matches(this.text, this.arr[i].retezec) ) {
                                        System.out.println("spatne");
                                } else {
                                        System.out.println("spravne");
                                }
                        }
                        if (type == "e") {
                                if ( Pattern.matches(this.email, this.arr[i].retezec) ) {
                                        System.out.println("spravne");
                                } else {
                                        System.out.println("spatne");
                                }
```

```
                        }
                        if (type == "u") {
                                if ( Pattern.matches(this.url, this.arr[i].retezec) ) {
                                        System.out.println("spravne");
                                } else {
                                        System.out.println("spatne");
                                }
                        }
                }
        }
}

class Validace {

        public static void main(String args[]){
                Formular form = new Formular(5);

                form.add("Toto je prosty text.", "t");

                form.add("pepa.jozef@hornidolni.ltd", "e");
                form.add("pepa.karel@dolni@horni.net", "e");

                form.add("htk://wwa.se;znam.cz/", "u");
                form.add("http://www.gmail.com/", "u");

                form.validovat();
        }
}
```

Zdrojový kód A.3: Zdrojový kód v jazyce PHP psaný zkušeným programátorem

```
#!/usr/bin/php -Cq
<?php
#konstanty
define("URL_REGEXP","/^((https?|ftps?|file|news|gopher):\\/\\/)?((?:(?:25[0-5]|2[0-4]\d|(?:(?:1\d)?|[1-9]?)\d)\.)
    {3}(?:25[0-5]|2[0-4]\d|(?:(?:1\d)?|[1-9]?)\d)|([0-9a-z]{1}[0-9a-z\\-]*\\.)*([0-9a-z]{1}[0-9a-z\\-]{0,56})\\.([a-z
    ]{2,6}|[a-z]{2}\\.[a-z]{2,6})(:[0-9]{1,4})?)(\\/?|.*)$/i");
define("EMAIL_REGEXP","/[-a-z0-9!#$%&\\'*+\/=?^_`{|}~.]+@([.]?[a-zA-Z0-9_\/-])*/");
#funkce
function append($array, $typ, $value) {
        $array[$typ][sizeof($array[$typ])] = $value;
        return $array;
}
function ClearHTML($sInput) {
        $sInput = preg_replace('/</','␣<',$sInput);
        $sInput = preg_replace('/>/','>␣',$sInput);
        $sInput = strip_tags($sInput,$AllowedTags);
        if($sTyp=="s") $sInput=HTMLSpecialChars($sInput);
        $sInput = preg_replace('/[\n\r\t\f]/','␣',$sInput);
        if($sTyp=="s") $sInput = str_replace('"','&quot;',$sInput);
        $sInput = Trim(preg_replace('/␣␣/','␣',$sInput));
        return $sInput;
}
function testInput($sTyp,$sInput) {
        $navrat=false;
        $sInput=Trim($sInput);
        if(!(Empty($sInput) and !$iPovinne)) {
                switch ($sTyp) {
                        case "e":
                                if(preg_match(EMAIL_REGEXP,$sInput))
                                        $navrat=true;
                                break;
                        case "u":
                                if(preg_match(URL_REGEXP,$sInput))
                                        $navrat=true;
                                break;
                        case "s":
                                $x=ClearHTML($sTyp,$sInput);
                                if(!Empty($x)) $navrat=true;
                                break;
                }
        }
        return $navrat;
}
function validate($data) {
        $typy = array(0 => "s","e","u");
        for ($k=0; $k<sizeof($typy); $k++) {
                $type=$typy[$k];
                for ($i=0; $i<sizeof($data[$type]); $i++) {
                        $vstup=$data[$type][$i];
                        echo $type;
```

```
                        echo "\t";
                        echo "Hodnota:␣\"".$vstup."\"";
                        //vsechny vstupy bereme jako povinne
                        if (testInput($type,$vstup)) {
                                echo "␣je␣validni.\n";
                        }else{
                                echo "␣neni␣validni!\n";
                        }
                }
        }
}
#vlozeni hodnot
// validni data
$data = array('s' => array(), 'e' => array(), 'u' => array());
$data = append($data, "s", "Toto␣je␣prosty␣text.");
$data = append($data, "e", "pepa.jozef@hornidolni.ltd");
$data = append($data, "u", "http://www.gmail.com/");
//invalidi data
$data = append($data, "e", "pepa.karel@dolni@h\orni.net");
$data = append($data, "u", "htk://seznam.cz/");
validate($data);
?>
```

Zdrojový kód A.4: Zdrojový kód v jazyce PHP psaný začátečníkem

```
#!/usr/bin/php -Cq
<?php

# zakladni struktury
$data = array('text' => array(), 'email' => array(), 'url' => array());
$messages = array('valid', 'invalid');

# fce
function append($pole, $typ, $val) {
        $index = count($pole[$typ]);
        $pole[$typ][$index] = $val;
        return $pole;
}

function validovat($typ, $val) {
        if ($val == "") { return 1; }

        switch ($typ) {
                case "text":
                        return preg_match('/<(\/|\?)*[A-Za-z0-9␣]*(\/|\?)*>/', $val);
                        break;
                case "email":
                        return !preg_match('/[A-Za-z0-9\-_]+\@[A-Za-z0-9\-_]+\.[a-z]+$/', $val);
                        break;
                case "url":
                        return !preg_match('/(http:\/\/)?(www\.)?[A-Za-z]+\.[a-z]+$/', $val);
```

```
                        break;
        }
}

#nacist data
// validni data
$data = append($data, 'text', "Toto je prosty text.");
$data = append($data, 'email', "pepa.jozef@hornidolni.ltd");
$data = append($data, 'url', "http://www.gmail.com/");
//invalidi data
$data = append($data, 'email', "pepa.karel@dolni@horni.net");
$data = append($data, 'url', "htk://wwa.se;znam.cz/");
#projedem pole
foreach ($data as $key => $oink) {
        echo "$key:\n";
        foreach ($data[$key] as $val) {
                echo "\t* $val "; echo $messages[validovat($key, $val)]; echo "\n";
        }

}

?>
```

Zdrojový kód A.5: Zdrojový kód v jazyce Python psaný zkušeným programátorem

```
#!/usr/bin/env python
import re

class Form:
        data = {'s': [],'e': [], 'u': []}
        def validate(self):
                def testInput(sTyp,sInput):
                        re_url=re.compile("^((ht|f)tp(s?)\:\/\/|~/|/)?([\w]+:\w+@)?([a-zA-Z]{1}([\w\-]+\.)+([\w]{2,5}))(:[\d
                                ]{1,5})?(((/?\w+/)+|/?)(\w+\.[\w]{3,4})?((\?\w+(=\w+)?)(&\w+(=\w+)?)*)?")
                        re_email=re.compile("^[a-zA-Z0-9._%-]+@[a-zA-Z0-9._%-]+\.[a-zA-Z]{2,6}$")
                        navrat=0
                        sInput.strip()
                        if sInput:
                                if sTyp=="u":
                                        if re_url.match(sInput) != None:
                                                navrat=1
                                if sTyp=="e":
                                        if re_email.match(sInput) != None:
                                                navrat=1
                                if sTyp=="s":
                                        navrat=1
                        return navrat
                for typ in self.data.keys():
                        for value in self.data[typ]:
                                vysledek=testInput(typ,value)
                                if vysledek:
                                        status=" je validni!"
                                else:
                                        status=" neni validni!"
                                print typ + "\tHodnota: " + value + status
        def readInput(self):
                #validi data
                self.data["s"].append("Toto je prosty text.");
                self.data["e"].append("pepa.jozef@hornidolni.ltd");
                self.data["u"].append("http://www.gmail.com/");
                #invalidi data
                self.data["e"].append("pepa.karel@dolni@horni.net");
                self.data["u"].append("htk://wwa.se;znam.cz/");
if ___name___ == "___main___":
        formular= Form()
        formular.readInput()
        formular.validate()
```

Zdrojový kód A.6: Zdrojový kód v jazyce Python psaný začátečníkem

```
#!/usr/bin/env python

# validace.py - validace vstupu z cli
# by Tilya Elerrama
# under GNU GPL v2

import re

class Formular:
        # struktury
        data = {'text': [],'email': [], 'url': []}
        # 0 - neplatne, 1 - v poradku, pripadne dalsi
        odpovedi = ['␣invalid␣', '␣ok␣']
        def validovat_text(self,val):
                if val != "" and re.search(r'<(\/|\?)*[A-Za-z0-9␣]*(\/|\?)*>', val): return 1
                else: return 0
        def validovat_email(self,val):
                if re.match(r'[A-Za-z0-9\-_]+\@[A-Za-z0-9\-_]+\.[a-z]+$', val): return 1
                else: return 0
                return 0
        def validovat_url(self,val):
                # [http://][www.]neco.com[/]
                #reg = re.compile(r'(http://)?(www\.)?[A-Za-z]+\.[a-z]$')
                if re.match(r'(http://)?(www\.)?[A-Za-z]+\.[a-z]+$',val): return 1
                else: return 0
        def ___init___(self):
                # hlavicka upravit
                if ___name___ == "___main___":
                        print """============================
VALIDACE␣v0.1
by␣Tilya␣Elerrama
under␣GNU␣GPL␣v2
----------------------------
pouziti:␣validace.py
(ano,␣zadne␣dalsi␣parametry)
============================
"""
        def nacist(self):
                #validni data
                self.data['text'].append("Toto␣je␣prosty␣text.");
                self.data['email'].append("pepa.jozef@hornidolni.ltd");
                self.data['url'].append("http://www.gmail.com/");
                #invalidi data
                self.data['email'].append("pepa.karel@dolni@horni.net");
                self.data['url'].append("htk://wwa.se;znam.cz/");
        def validovat(self):
                # projdeme slovnik a pro kazdy typ zavolame jeho funkci (getattr)
                # vypisujeme
                # zjistime klice a jdeme podle nich
                for klic in self.data.keys():
```

```
                print klic + ":"
                for val in self.data[klic]:
                        # validace
                        a = getattr(self, "validovat_" + klic)
                        print "\t* " + val + "\t",
                        print self.odpovedi[a(val)]



if __name__ == "__main__":
        form = Formular()
        form.nacist()
        form.validovat()
```

Zdrojový kód A.7: Zdrojový kód v jazyce Ruby psaný zkušeným programátorem

```
#!/usr/bin/env ruby

class Form
	@@data = {"s" => [], "e" => [], "u" => []}

	def testInput(sTyp, sInput)
		re_url=/(^$)|(^(http|https):\/\/[a-z0-9]+([\-\.]{1}[a-z0-9]+)*\.[a-z]{2,5}(([0-9]{1,5})?\/.*)?$)/ix
		re_email=/\A([\w\.\-\+]+)@((?:[-a-z0-9]+\.)+[a-z]{2,})\z/i
		navrat=0
		sInput.strip
		if sInput !=""
			regexp = re_url if sTyp== "u"
			regexp = re_email if sTyp== "e"
			return 1 if sTyp== "s"
			if sInput = ~ regexp
				navrat=1
			end
		end
		return navrat
	end

	def validate
		@@data.keys.each do |sTyp| # mame list
			@@data[sTyp].each do |sInput|
				vysledek=testInput(sTyp, sInput)
				if vysledek == 1
					status="␣je␣validni!"
				else
					status="␣neni␣validni!"
				end
				print sTyp + "\tHodnota:␣" + sInput + status + "\n"
			end
		end
	end

	def readInput
		#validi data
		@@data["s"].push("Toto␣je␣prosty␣text.");
		@@data["e"].push("pepa.jozef@hornidolni.ltd");
		@@data["u"].push("http://www.gmail.com/");
		#invalidi data
		@@data["e"].push("pepa.karel@dolni@horni.net");
		@@data["u"].push("htk://wwa.se;znam.cz/");
	end
end

formular = Form.new
formular.readInput
formular.validate
```

Zdrojový kód A.8: Zdrojový kód v jazyce Ruby psaný začátečníkem

```
#!/usr/bin/env ruby

# validace.rb - validace vstupu z cli
# by Tilya Elerrama
# under GNU GPL v2


# trida

class Formular
        def initialize
                #promenne
                @data = {'url' => [], 'email' => [], 'text' => []} # kontejner na data
                @message = ['spravne', 'spatne'] # popelnice na vysledek

                #regexpy
                @text = %r{<(\/|\?)*[A-Za-z0-9 ]*(\/|\?)*>}
                @email = %r{[A-Za-z0-9\-_]+\@[A-Za-z0-9\-_]+\.[a-z]+$}
                @url = %r{(http://)?(www\.)?[A-Za-z]+\.[a-z]+$}

                #hlavicka
                puts"============================
VALIDACE
by Tilya Elerrama
under GNU GPL v2
----------------------------
pouziti: validace.rb
(ano, zadne dalsi parametry)
============================\n\n"
                #validi data
                @data["text"].push("Toto je prosty text.");
                @data["email"].push("pepa.jozef@hornidolni.ltd");
                @data["url"].push("http://www.gmail.com/");
                #invalidi data
                @data["email"].push("pepa.karel@dolni@horni.net");
                @data["url"].push("htk://wwa.se;znam.cz/");
        end

        def print_dict
                @data.keys.each do |key|
                        puts key + ":"
                        @data[key].each do |item|
                                print "\t* " + item + " => "
                                puts @message[validovat(item, key)]
                        end
                end
        end

        def validovat(string,typ)
                #test na prazdny string, vzdycky spatne
```

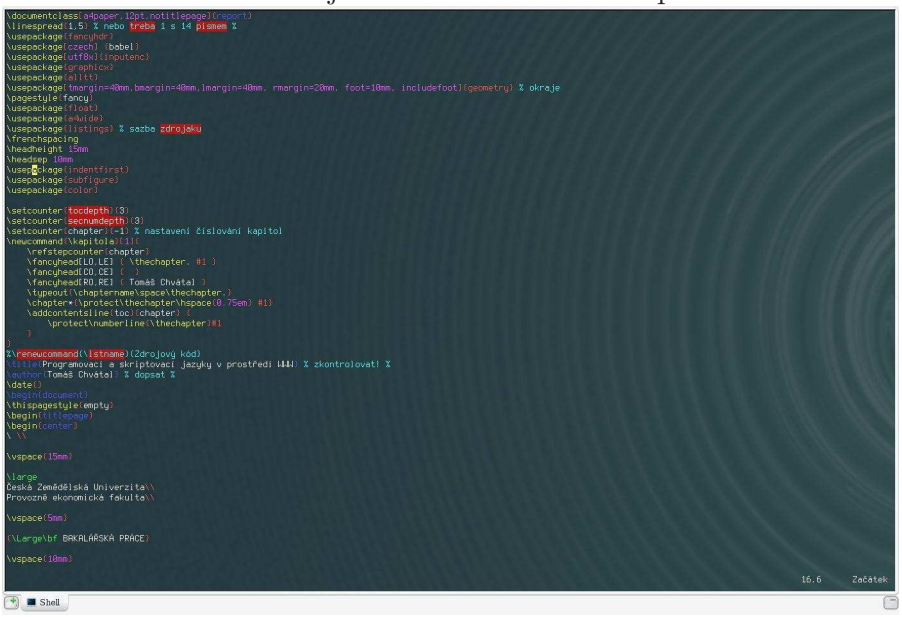
```
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣return␣1␣if␣string␣==␣""

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣#priradime
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣pat␣=␣@text␣if␣typ␣==␣"text"
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣pat␣=␣@email␣if␣typ␣==␣"email"
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣pat␣=␣@url␣if␣typ␣==␣"url"

␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣oink␣=␣string.scan(pat).length
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣if␣typ␣!=␣"text"
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣return␣oink␣>␣0␣?␣0␣:␣1
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣else
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣return␣oink␣>␣0␣?␣1:␣0
␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣end
␣␣␣␣␣␣␣␣end
end

form␣=␣Formular.new
form.print_dict
```

Obrázek A.1: Ukázka zdrojového kódu bakalářské práce v editoru ViM



Obrázek A.2: Ukázka zdrojového kódu bakalářské práce v editoru Kate