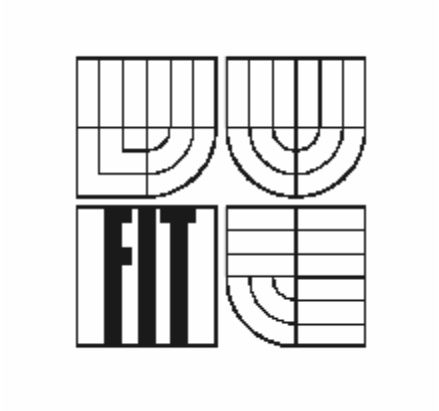
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ROČNÍKOVÝ PROJEKT

2006 **Kmenta Karel**

Zadání

Kvantifikace podobnosti programových souborů

- Prostudujte problematiku porovnávání textových souborů a jednorozměrných signálů. Zaměřte se na problémy s porovnáváním zdrojových souborů.
- Identifikujte problémy, které se mohou při porovnávání zdrojového kódu vyskytnout.
  Navrhněte algoritmus pro porovnávání zdrojových souborů zvoleného, běžně používaného programovacího jazyka.
- Implementací ověřte navržený algoritmus. Zvažte možnost vytvoření modulárního systému, který umožní přidávat moduly pro porovnávání zdrojového kódu dalších programovacích jazyků. Demonstrujte správnost navrženého algoritmu na vhodně zvolených testovacích datech.
- Zhodnoťte dosažené výsledky a navrhněte další možné směry vývoje.

Prohlášení

Prohlašuji, že jsem tento ročníkový projekt vypracoval samostatně pod vedením Ing. Davida Martinka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Abstrakt

Tento projekt se zabývá problematikou porovnávání zdrojových souborů, různými metodami pro porovnávání a navržením algoritmu pro porovnání zdrojových souborů programovacího jazyka C. Projekt má sloužit jako pomůcka při odhalování plagiátorství řešených projektů a dalších prací.

Klíčová slova

Lexikální analýza, LL-gramatika, syntaktická analýza, konvoluce, plné textové porovnání

Abstract

This project concerns about problems of comparing of source files, various methods for comparing and design of algorithm for comparing of source files of programming language C. Project services as aid to detection of plagiarism solving projects and by other schoolworks.

Keywords

Lexikal analysis, LL-grammar, syntax analysis, convolution, full-text comparing

Obsah

1. Úvod

Tato práce čtenáře obeznámí s problematikou kvantifikace podobnosti programových souborů.

Čtenář se dozví více o úrovních podobnosti zdrojových souborů, jež rozlišuji do čtyř kategorií. Jakým způsobem, pozmění-li potenciální tvůrce plagiátu obsah zdrojového kódu, určuji, do které z kategorií, jako je *shoda*, *podobnost*, *na hraně rozpoznatelnosti* a *odlišnost*, jej zařadím.

Dále čtenáře provede analýzou metod pro řešení podobnosti programových souborů, jako je *plné textové porovnání*, *porovnání pomocí konvoluce* či *porovnání na základě syntaktické analýzy*. U jednotlivých metod je také navíc rozebráno, pro kterou z úrovní podobnosti zdrojových souborů, je daná metoda použitelná.

Po té je čtenář seznámen, jak jsem při řešení tohoto problému postupoval. Co mě vedlo k tomu, abych jako první krok odstranil poznámky z porovnávaných zdrojových souborů a následně využil metody, jež by nebyla před odstraněním poznámek prakticky použitelná. Jakým způsobem jsem dospěl k vytvoření tabulky symbolů a jaké prostředky jsem k její tvorbě použil. Jak jsem postupoval při výpočtu pravděpodobnosti shody či typové podoby u globálních proměnných a jakých jsem k tomu využil vzorců. Podobně je uveden také postup při vyhodnocování podobnosti funkcí z porovnávaných souborů.

Nakonec jsou uvedeny ukázky, jaké výsledky tento projekt, při zpracování *shodných*, *podobných*, *na hraně rozpoznatelnosti* i *odlišných* porovnávaných souborů, udává. Spolu s poznatky získané vypracováním projektu jsou tyto výsledky okomentovány a na jejich základě je předložen návrh dalšího možného směru vývoje.

2. Analýza

Při porovnávání zdrojových souborů je velmi komplikované určit, kde je hranice rozpoznatelnosti. Nejprve jsem si tedy promyslel úrovně podobnosti, jež po porovnávání zdrojových souborů vyžaduji.

Porovnávané soubory se na té nejnižší kategorii mohou lišit např. jen odlišnými poznámkami. Tyto poznámky při použití metody plného textového porovnání výsledek značně zkreslují. Vzhledem k tomu, že poznámky do funkčnosti programu nijak nezasahují, získám jejich odstraněním lepší výchozí pozici pro porovnávání a mám zajištěno, že metoda plného textového porovnání bude u souborů lišících se pouze poznámkami dostatečně průkazná.

Půjdu-li postupně v možných odlišnostech zdrojových souborů, pak dalším bodem jsou přejmenované globální proměnné, názvy funkcí, jejich parametry i lokální proměnné. Pokud osoba, jež má v úmyslu přepracovat cizí projekt, provede toto přejmenování, je metoda plného textové porovnání nevyhovující. Pro odhalení takového způsobu plagiátorství je již nutno vzít silnější nástroj na porovnávání. Takovým nástrojem je metoda využívající syntaktické analýzy, kde s pomocí lexikálního analyzátoru postupně čtu a vkládám slova z porovnávaných zdrojových souborů do tabulek symbolů. Jejich položky pak snadno porovnám a určím, zda mohlo dojít k přejmenování názvů proměnných a funkcí.

Metoda s pomocí syntaktické analýzy je výhodná i v případě, kdy se ve zdrojových souborech vyskytují globální proměnné a funkce v různém pořadí. Zde je již zapotřebí i odkazů na jednotlivé funkce, aby je bylo možné porovnávat, přestože se vyskytují v různých částech zdrojových souborů.

Pokud jsou funkce a proměnné nejen přejmenovány a zpřeházeny, ale navíc i modifikovány, tzn. že struktura příkazů bude mít jinou podobu, pak je již velmi obtížné určit, zda se jedná o plagiátorskou činnost či natolik modifikovatelnou verzi, kterou již lze pokládat za tvůrčí úsilí jedné z osob, nebo zda se jedná o naprosto samostatné dílo.

Úplně jiný pohled na porovnání zdrojových souborů má metoda s pomocí konvoluce. Zde se využívá poznatků o jednorozměrných signálech, kdy je brán řádek z jednoho zdrojového souboru a z opačné strany je k němu přikládán řádek z druhého porovnávaného zdrojového souboru. Výsledkem je křivka, která určuje, do jaké míry si jsou porovnávané řádky podobné.

2.1 Úrovně podobnosti zdrojových souborů

Úrovně podobnosti zdrojových souborů můžeme prakticky klasifikovat do čtyř skupin. A to:

- Shodné (2.1.1)
- Podobné (2.1.2)
- Na hraně rozpoznatelnosti (2.1.3)
- Odlišné (2.1.4)

2.1.1 Shodné

Zdrojové soubory považujeme za shodné tehdy, pokud jsou od začátku až do konce při plném textovém porovnání (znak po znaku) totožné.

```
int x, y, z;

int main(void) {
  int i;
  float a, b;

  i = 1;
  while(i <= 10) {
    a = 1.0 / i;
    printf("%f\n", a);
    i++;
  }
  .
  .
```

```
int x, y, z;

int main(void) {
  int i;
  float a, b;

  i = 1;
  while(i <= 10) {
    a = 1.0 / i;
    printf("%f\n", a);
    i++;
  }
  .
  .
```

obr.1 – ukázka zdrojového kódu *shodných* souborů

2.1.2 Podobné

Do této skupiny patří porovnávané soubory, u nichž byla struktura příkazů a těl jednotlivých funkcí velmi podobná a odlišnost byla dána jen přejmenováním názvů funkcí, globálních a lokálních proměnných.

```
int x, y, z;

int main(void) {
  int i;
  float a, b;

  i = 1;
  while(i <= 10) {
    a = 1.0 / i;
    printf("%f\n", a);
    i++;
  }
  .
  .
```

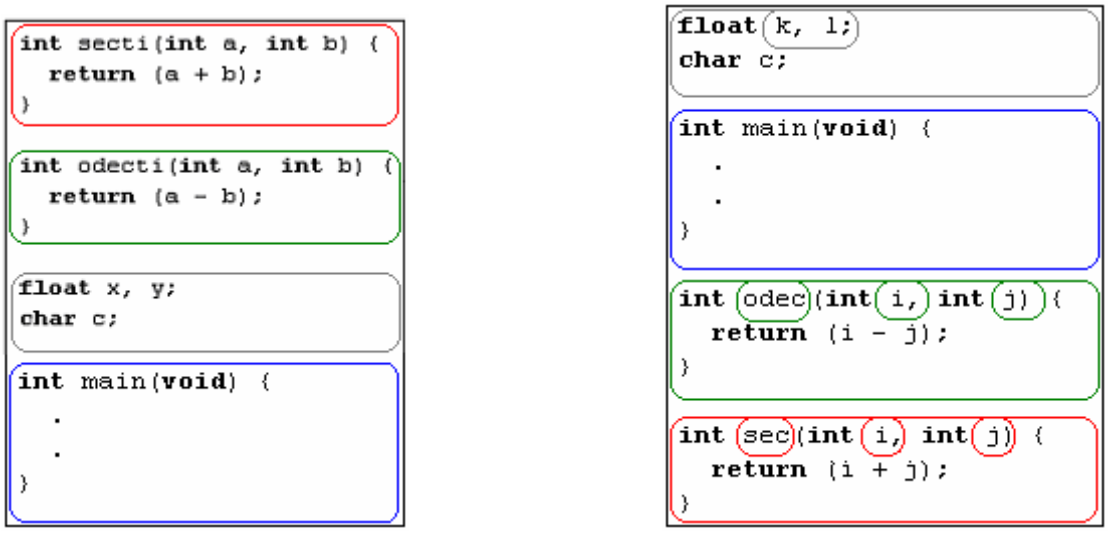
```
int a, b, c;

int main(void) {
  int j;
  float x, y;

  j = 1;
  while(j <= 10) {
    x = 1.0 / j;
    printf("%f\n", x);
    j++;
  }
  .
  .
```

obr.2 – ukázka zdrojového kódu *podobných* souborů

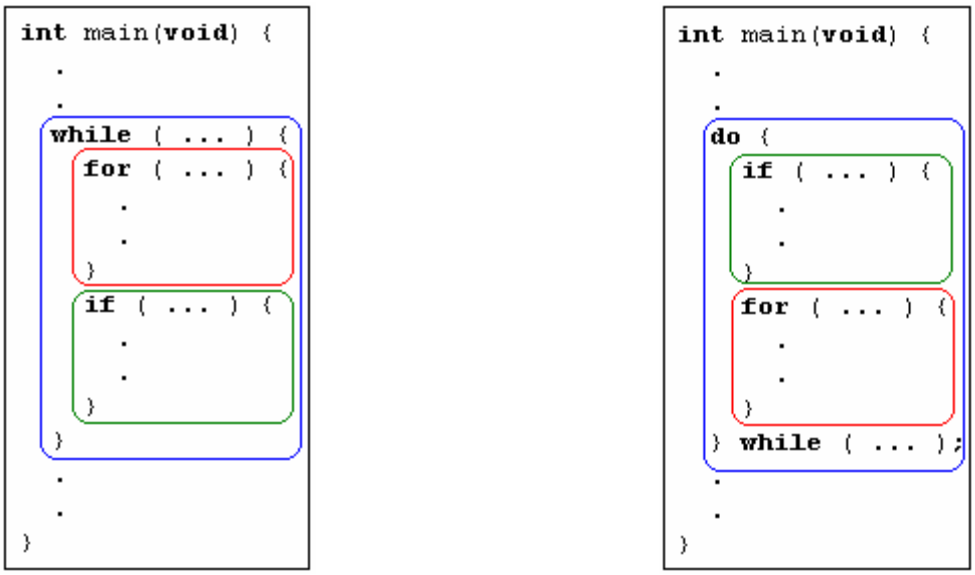
Mimo jiné sem můžeme zahrnout i takový případ, kdy nejenže mohou být názvy funkcí a proměnných přejmenovány, ale navíc se v porovnávaných souborech mohou vyskytovat i v jiném pořadí.



obr.3 **–** ukázka zdrojového kódu *podobných* souborů

2.1.3 Na hraně rozpoznatelnosti

Zde se již jedná o vyšší stupeň při stanovení podobnosti. Odlišné názvy funkcí a proměnných, různá stavba těl funkcí a struktura příkazů je v takovém stavu, že již není snadné rozpoznat, zda se jedná o přepracování jednoho ze zdrojových souborů či o na sobě nezávisle vypracované zdrojové soubory.



obr.4 **–** ukázka zdrojového kódu souborů *na hraně rozpoznatelnosti*

2.1.4 Odlišné

Tato skupina označuje porovnávané soubory za odlišné v případě, že jsou svou strukturou a stavbou natolik různé, že o nezávislosti vytvoření zdrojových souborů na sobě není pochyb.

```
int x, y, z;

int main(void) {
  int i;
  float a, b;

  i = 1;
  while(i <= 10) {
    a = 1.0 / i;
    printf("%f\n", a);
    i++;
  }
  .
  .
```

```
int secti(int a, int b) {
  return (a + b);
}

int odecti(int a, int b) {
  return (a - b);
}

float x, y;
char c;

int main(void) {
  .
  .
}
```

obr.5 – ukázka zdrojového kódu *odlišných* souborů

2.2 Porovnání zdrojových souborů

Na porovnání zdrojových souborů existuje několik metod. Ač budeme porovnávat těmito různými metodami stejné dvojice zdrojových souborů, každá metoda nám vyhodnotí porovnání různě, a to podle jejího způsobu zpracování.

Metody sloužící k porovnávání jsou:

- Plné textové porovnání (2.2.1)
- Konvoluce (2.2.2)
- Porovnání pomocí syntaktické analýzy (2.2.3)

2.2.1 Plné textové porovnání

Pod pojmem *plné textové porovnání* je skryta metoda, jež porovnává zdrojové soubory znak po znaku od začátku až po konec těchto souborů. Z toho je zřejmé, že metoda je použitelná jen pro případ odhalení naprosté totožnosti zdrojových souborů. Pokud je jeden z porovnávaných souborů přepracovanou verzí toho druhého, a to takovým způsobem, že jsou přejmenovány názvy funkcí i proměnných, popřípadě že těla funkcí a struktura příkazů jsou navíc zcela zpřeházeny, je tato metoda nevyhovující. U této metody tak prakticky dostáváme výsledek dvojího typu:

- zdrojové soubory jsou *shodné*
- zdrojové soubory jsou *odlišné*

Použitelnost této metody je tak na základě jejího vyhodnocení do značné míry omezena. Na druhou stranu její princip spolu v kombinaci s jinou metodou už může uspokojit naše požadavky.

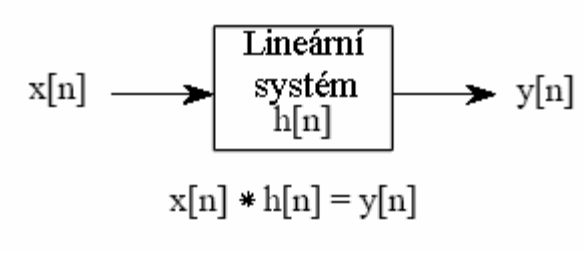
2.2.2 Porovnání pomocí konvoluce

Konvoluce je formální matematická operace, tak jako třeba násobení, sčítání nebo integrování. Při sčítání máme například dvě čísla a jejich součtem získáme třetí. Podobně je tomu u konvlouce, kde máme dva signály a jejich sloučením získáme signál třetí. Vstupní signál se mění na výstupní následovně:

- vstupní signál je rozložen na řadu impulsů, každý impuls může být zapsán jako funkce delta
- na výstupu se každý impuls zapíše jako impulsní charakteristika
- výstupní signál vzniká složením výstupních impulsů

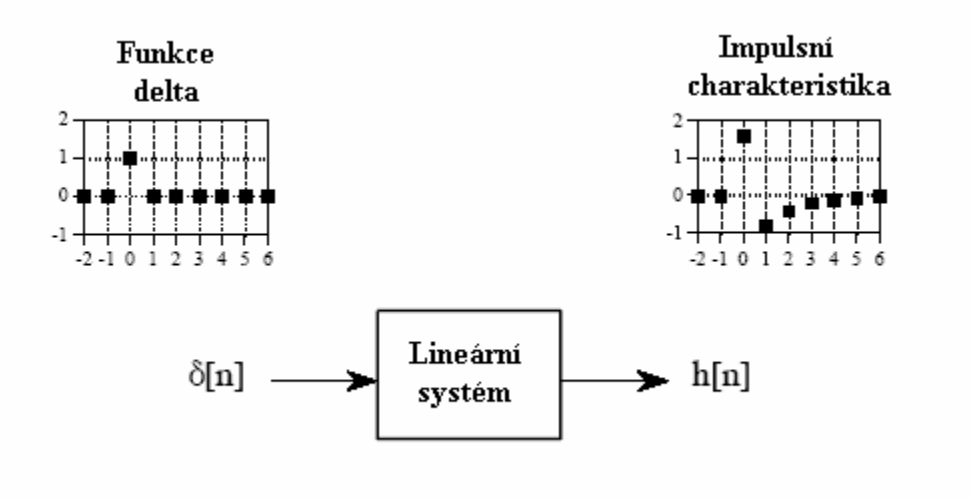
Takto vzniklá impulsní charakteristika má různé názvy, jako například *filtr kernel* (jádro filtru), *konvoluční kernel* (jádro konvoluce), nebo prostě jen *kernel*.

Na následujícím obrázku je uveden příklad použití konvoluce lineárním systémem. Na vstup lineárního systému s impulsní charakteristikou h[n] je přiváděn vstupní signál x[n], výsledkem je výstupní signál y[n], který vznikl konvolucí.



obr.6 – použití konvoluce lineárním systémem

Důležitým termínem je *funkce delta*. Je to normalizovaný impuls, což znamená, že hodnota vzorku 0 je rovna 1, pokud se všechny ostatní vzorky rovnají nule. *Funkce delta* je značena písmenem řecké abecedy malé delta. Této funkci se také často říká *jednotkový impuls*.



obr.7 – impulsní charakteristika po průchodu *funkce delty* lineárním systémem

Konvoluce se tak může využít např. při porovnávání po řádcích, kdy oba porovnávané řádky rozložíme na posloupnost impulsů a po té tyto řádky skonvolujeme. To, do jaké míry si jsou porovnávané řádky podobné určuje křivka, jež je výsledkem konvoluce.

2.2.3 Porovnání pomocí syntaktické analýzy

Tato metoda je velmi silným nástrojem pro porovnávání zdrojových souborů. Vzhledem k tomu že vychází ze syntaktické analýzy, je možné ji použít i v takových případech, kdy jsou těla funkcí a struktura příkazů ve zdrojových souborech zpřeházeny. Využívá toho, že na základě pravidel *LL-gramatiky* pro daný jazyk, jsou vytvořeny množiny *first(), follow(), empty()* a *predict()*, s jejichž pomocí je pak sestavena *LL-tabulka*. K realizaci u této metody potřebuji mimo jiné i lexikální analyzátor, jež ze vstupního souboru čte znaky reprezentující zdrojový program a z těchto znaků vytváří symboly programu, což je již vhodná forma pro zpracování. Výstup z lexikálního analyzátoru je předáván syntaktické analýze, jež s pomocí vytvořené *LL-tabulky* a na základě pravidel *LL-gramatiky* zvoleného jazyka vyhodnotí, jak dál postupovat. Výstupem syntaktické analýzy je abstraktní derivační strom.

Pro porovnávání zdrojových souborů je pak sestavena hloubka abstraktního derivačního stromu na základě požadavků, jež určují do jaké míry se má vyhodnocení podobnosti provézt. Samotný výpočet pravděpodobnosti podoby porovnávaných souborů je pak velmi variabilní. Vzhledem k tomu, že se lze ve stromě poměrně dobře dynamicky pohybovat, je sestavení vyhodnocujících podmínek a rovnic individuální záležitostí.

3. Implementace

Vzhledem k tomu, že existuje několik metod na porovnávání zdrojových souborů, vybral jsem si na základě mých požadavků metodu využívající syntaktické analýzy v kombinaci s plným textovým porovnáváním. To umožňuje odhalit nejen přejmenování názvů proměnných a funkcí, ale i jejich zpřeházení ve zdrojových souborech. Algoritmus je navržen pro porovnávání zdrojových souborů programovacího jazyka C. Projekt byl řešen v operačním systému WindowsXP v prostředí Builder C++ a v Linuxu přeložen překladačem g++. Názvy vstupních porovnávaných souborů jsou zadávány jako argumenty programu.

3.1 Odstranění poznámek

Abych ponechal zdrojové soubory nepozměněny, vytvořil jsem nejprve soubory pomocné. S nimi pracuji až do výsledného porovnání. Jako další úkon jsem zvolil odstranění poznámek z obou porovnávaných zdrojových souborů. Poznámky by výsledné porovnání mohly do značné míry ovlivnit a tak jejich odstraněním získám soubory obsahující čistě jen zdrojový kód.

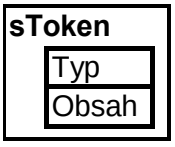
3.2 Test na shodu

Dalším bodem -po odstranění poznámek- je test, zda se nyní zdrojové kódy shodují. K tomuto porovnání je použita metoda *plného textového porovnání*. Výsledkem tohoto testu může být jen shoda nebo neshoda porovnávaných pomocných souborů. Jestliže výsledek porovnání udává shodu, pak je toto oznámeno uživateli. Další porovnávání již není zapotřebí a program je tak ukončen.

3.3 Vytvoření tabulky symbolů

V případě, že test na shodu udá jako svůj výsledek neshodu porovnávaných souborů, dochází k tvorbě tabulek symbolů pro oba pomocné soubory. Na tvorbu tabulek symbolů je zapotřebí lexikální analyzátor. Ten čte ze zadaného souboru po slovech. Jeho výstupem je struktura obsahující typ přečteného slova a obsah tohoto slova. Struktura má tedy podobu uvedenou na obrázku a může vracet typ:

- identifikátor
- klíčové slovo
- klíčové slovo typu
- operátor
- integer
- float
- řetězec
- znak



obr.8 – struktura vystupující z lexikálního analyzátoru

Slova z lexikálního analyzátoru zpracovává syntaktická analýza. V případě že je syntaktické analýze předána globální proměnná nebo funkce, je tato globální proměnná, příp. funkce, uložena do tabulky. Ta má následující podobu:

Název [string]	Typ [string]	GP / FCE [int]	Pozice [long]	Stav [bool]

obr.9 – tabulka sloužící k uložení globálních proměnných a funkcí ze zdrojového souboru

Význam jednotlivých položek tabulky je:

- Název – název globální proměnné, příp. funkce
- Typ – typ globální proměnné, příp. funkce
- GP/FCE – udává, zda se jedná o globální proměnnou nebo funkci
- Pozice – pokud je zpracovávána funkce, je zde uložena pozice, na kterém místě se tato funkce v daném souboru nachází
- Stav – tato položka označuje, zda byla spočítaná podobnost funkce již vybrána

3.4 Výpočet pravděpodobnosti u globálních proměnných

Vyhodnocení porovnání globálních proměnných je dvojího druhu. První způsob kontroluje, zda se globální proměnné shodují ve svých názvech i typech. Druhý způsob názvy funkcí ignoruje a zaměřuje se pouze na typy jednotlivých globálních proměnných.

3.4.1 Pravděpodobnost shody globálních proměnných

K vypočítání pravděpodobnosti shody globálních proměnných je použita pomocná tabulka, jež byla vytvořena s pomocí tabulek symbolů. Tabulka má následující podobu:

Název GP1 [string]	Název GP2 [string]	Shoda [bool]

obr.10 – tabulka sloužící pro vyhodnocení shody globálních proměnných

Význam jednotlivých položek tabulky:

- Název GP1 – název globální proměnné z prvního pomocného souboru
- Název GP2 – název globální proměnné z druhého pomocného souboru
- Shoda – udává, zda jsou tyto globální proměnné shodné

Pro každou globální proměnnou z prvního pomocného souboru testuji, zda nedošlo ke shodě s některou z globálních proměnných z druhého pomocného souboru. V tabulce se tedy každá globální proměnná prvního pomocného souboru vyskytuje tolikrát, kolik globálních proměnných je obsaženo v druhém pomocném souboru. Pro každou dvojici globálních proměnných je vyhodnocen stav, zda se shodují ve svých názvech i typech. Pokud výsledek udává shodu, je položka *Shoda* nastavena jako *true*.

Výsledná pravděpodobnost shody globálních proměnných je spočítána dle vzorce:

$$P_V = \frac{2 * p_{Sh}}{cp} * 100 \ [\%], \text{ kde:} \qquad (1)$$

- P_V - výsledná pravděpodobnost
- p_{Sh} - počet shodných dvojic
- cp - celkový počet globálních proměnných

3.4.2 Pravděpodobnost typové podobnosti globálních proměnných

Při výpočtu této pravděpodobnosti jsou názvy globálních proměnných zcela ignorovány. Pro každý typ proměnné je spočítán počet výskytů globálních proměnných tohoto typu v jednotlivých souborech a vypočítána pravděpodobnost právě pro tento typ.

$$P = \frac{cp_1}{cp_2} * 100 \ [\%], \text{ kde:} \qquad (2)$$

- P - pravděpodobnost pro jednotlivé typy globálních proměnných
- cp_1 - celkový počet globálních proměnných daného typu z prvního pomocného souboru
- cp_2 - celkový počet globálních proměnných daného typu z druhého pomocného souboru

Jakmile jsou spočítány pravděpodobnosti pro všechny typy, je spočítána pravděpodobnost výsledná.

$$P_V = \frac{\sum P_i}{i} * 100 \ [\%], \text{ kde:} \qquad (3)$$

- P_V - výsledná pravděpodobnost
- P_i - spočítaná pravděpodobnost dle vzorce (2) pro jednotlivé typy, kde i udává počet těchto spočítaných pravděpodobností

3.5 Výpočet pravděpodobnosti u funkcí

Podobně jako u výpočtu pravděpodobnosti globálních proměnných je i zde výsledek dvojího druhu. První způsob vyhodnocuje podobnost funkcí na základě sekvencí slov obsažených ve funkcích. Druhý způsob určuje podobnost funkcí na základě posloupnosti klíčových slov v těle každé z funkcí.

Protože osoba, jež je potenciálním tvůrcem plagiátu, může funkci přidat či ubrat nějaký parametr, zpřeházet její lokální proměnné či jiné úpravy, bylo před samotným vyhodnocením pravděpodobností nezbytné provézt jisté úpravy.

Pokud by byly pouze přejmenovány lokální proměnné, pak by jistě postačilo nahrazení těchto proměnných vhodně zvolenou náhradou. V případě, že jsou lokální proměnné funkce i v jiném pořadí, nemusí toto nahrazení stačit, protože náhradu, jež jsem učinil v prvním porovnávaném souboru, použiji ve druhém souboru nechtěně na jiném místě. Proto jsem zvolil možnost, kdy všechny parametry a lokální proměnné dané funkce nahradím jejich typy.

Proto jsem vytvořil novou tabulku pro parametry a lokální proměnné, jež daná funkce obsahuje. S pomocí položky *pozice* v původní tabulce symbolů, se dostanu na začátek funkce, kterou budu porovnávat a provedu již zmíněné přejmenování v celém těle funkce. Stejné operace provedu i pro funkci z druhého pomocného souboru. Nyní se již mohu věnovat samotnému porovnávání funkcí.

3.5.1 Podobnost na základě sekvence slov

Podobně jako u porovnávání globálních proměnných je i zde každá funkce z prvního pomocného souboru porovnávána se všemi funkcemi z druhého pomocného souboru. Spočítané pravděpodobnosti jsou uloženy do tabulky, jež má podobu:

Název FCE1 [string]	Název FCE2 [string]	Podobnost [float]

obr.11 – tabulka sloužící pro vyhodnocení podobnosti funkcí na základě sekvence slov

Význam jednotlivých položek tabulky:

- Název FCE1 – název funkce z prvního pomocného souboru
- Název FCE2 – název funkce z druhého pomocného souboru
- Podobnost – vyhodnocená pravděpodobnost podoby obou funkcí

Jakmile je provedena náhrada proměnných v jednotlivých funkcích, jsou sekvence slov obou těl funkcí uloženy do front, jejichž položky se následně vyhodnocují a určují výslednou pravděpodobnost podoby funkcí. Tato pravděpodobnost je dána vzorcem (1), kde:

- P_V - výsledná pravděpodobnost
- p_{Sh} - počet shodných dvojic odpovídajících si položek z obou front
- cp - celkový počet položek obou front

Protože při vyhodnocování pravděpodobnosti jsou funkce porovnávány „každá s každou“, obsahuje tabulka na konci tohoto porovnání přebytečné položky. Pro určení, které funkce z prvního a z druhého porovnávaného souboru jsou si nejvíce podobné, je postupováno následovně. Vybere se položka tabulky s nejvyšší hodnotou v kolonce *Podobnost*. Následně jsou z tabulky přečteny názvy funkcí, u nichž byla tato nejvyšší podobnost vyhodnocena. V tabulkách symbolů pro oba porovnávané soubory se zkontroluje, zda je položka *Stav* u vybraných funkcí *false*, což znamená, že daná funkce ještě nebyla dána do žádné dvojice s některou z funkcí z druhého souboru. Pokud je tento požadavek splněn u obou funkcí, je jejich *Stav* změněn na *true* a další spočítané výsledky v tabulce s pravděpodobnostmi jsou u těchto funkcí ignorovány. Dokud se vykytují v tabulce s pravděpodobnostmi dvojice funkcí, jejichž položka *Stav* v tabulce symbolů je *false*, je tento postup opakován.

Na závěr jsou vypsány dvojice nejpodobnějších funkcí z obou porovnávaných souborů. Počet těchto vypisovaných dvojic závisí na nadefinované konstantě *SIZE*, jejíž hodnota je v hlavičkovém souboru *pr1.h* nastavena na cifru 10. Pokud je počet dvojic funkcí menší než tato konstanta, pak jsou vypsány všechny tyto dvojice. V opačném případě je výpis nejpodobnějších funkcí dán právě hodnotou konstanty *SIZE*.

3.5.2 Podobnost na základě sekvence klíčových slov

Zde je postupováno stejným způsobem jako u předchozího bodu (3.5.1). Rozdíl je v tom, že do front nejsou vkládány sekvence všech slov obsažených ve funkcích, ale vkládá se do nich jen posloupnost klíčových slov. Pokud tvůrce plagiátu přidá do těla funkce nějakou proměnnou nebo složitější výpočet nahradí více příkazy, ale strukturu ponechá zachovanou, má podobnost spočítaná tímto způsobem vyšší vypovídací hodnotu než spočítaná podobnost na základě sekvence všech slov. Pro nově spočítané pravděpodobnosti je přidána do tabulky položka právě pro tyto hodnoty.

Název FCE1 [string]	Název FCE2 [string]	Podobnost [float]	Podobnost_KW [float]

obr.12 – tabulka sloužící pro vyhodnocení podobnosti funkcí obohacena o položku s podobnostmi vyhodnocených na základě sekvence klíčových slov

3.6 Zobrazení výsledků

Jakmile je pravděpodobnost u globálních proměnných i u funkcí spočítána, je tento výsledek zobrazen a spolu s ním i další doplňující informace. Ty zobrazují celkový počet globálních proměnných, funkcí a klíčových slov obsažených v těle jednotlivých funkcí.

Na závěr uvádím několik ukázek, jaké vyhodnocení program při testování podobnosti udává.

Pro *shodné* soubory je výstup programu následující:

```
!!! Zdrojovy kod porovnavanych souboru je shodny. !!!
```

obr.13 – výstup programu při porovnání *shodných* souborů

Pro *podobné* soubory, u nichž byly zpřeházeny globální proměnné i funkce, je výsledné určení podobnosti následující:

```
---------------------------------
!  Pocet globalnich promennych  !
---------------------------------
V 1.souboru: 1
V 2.souboru: 1
---------------------------------
Shodnych dvojic: 100.00%
Typova podobnost: 100.00%

---------------------------------
!          Pocet funkci         !
---------------------------------
V 1.souboru: 5
V 2.souboru: 5
---------------------------------
! Nejpodobnejsi dvojice funkci  !
---------------------------------
```

```
Dle sekvence slov:
------------------
Z 1.souboru: odstran_neslova,  Pocet klicovych slov: 3
Z 2.souboru: odstran_neslova,  Pocet klicovych slov: 3
Podobnost: 100.00%

Z 1.souboru: vrat_slovo,  Pocet klicovych slov: 3
Z 2.souboru: vrat_slovo,  Pocet klicovych slov: 3
Podobnost: 100.00%

Z 1.souboru: pridej_slovo,  Pocet klicovych slov: 4
Z 2.souboru: pridej_slovo,  Pocet klicovych slov: 4
Podobnost: 100.00%

Z 1.souboru: tiskni,  Pocet klicovych slov: 2
Z 2.souboru: tiskni,  Pocet klicovych slov: 2
Podobnost: 100.00%

Z 1.souboru: main,  Pocet klicovych slov: 7
Z 2.souboru: main,  Pocet klicovych slov: 7
Podobnost: 100.00%


Dle sekvence klicovych slov:
----------------------------
Z 1.souboru: odstran_neslova,  Pocet klicovych slov: 3
Z 2.souboru: odstran_neslova,  Pocet klicovych slov: 3
Podobnost: 100.00%

Z 1.souboru: vrat_slovo,  Pocet klicovych slov: 3
Z 2.souboru: vrat_slovo,  Pocet klicovych slov: 3
Podobnost: 100.00%

Z 1.souboru: pridej_slovo,  Pocet klicovych slov: 4
Z 2.souboru: pridej_slovo,  Pocet klicovych slov: 4
Podobnost: 100.00%

Z 1.souboru: tiskni,  Pocet klicovych slov: 2
Z 2.souboru: tiskni,  Pocet klicovych slov: 2
Podobnost: 100.00%

Z 1.souboru: main,  Pocet klicovych slov: 7
Z 2.souboru: main,  Pocet klicovych slov: 7
Podobnost: 100.00%
```

obr.14 – výstup programu při porovnání *podobných* souborů

Pro *podobné* soubory, u nichž byly globální proměnné i funkce přejmenovány a zpřeházeny, je výsledné určení podobnosti následující:

```
---------------------------------
!  Pocet globalnich promennych  !
---------------------------------
V 1.souboru: 1
V 2.souboru: 1
---------------------------------
Shodnych dvojic: 0.00%
Typova podobnost: 100.00%

---------------------------------
!          Pocet funkci         !
---------------------------------
V 1.souboru: 5
V 2.souboru: 5
---------------------------------
! Nejpodobnejsi dvojice funkci  !
---------------------------------
```

```
Dle sekvence slov:
------------------
Z 1.souboru: main,  Pocet klicovych slov: 3
Z 2.souboru: main,  Pocet klicovych slov: 3
Podobnost: 99.25%

Z 1.souboru: vrat_slovo,  Pocet klicovych slov: 3
Z 2.souboru: vrat,  Pocet klicovych slov: 3
Podobnost: 99.02%

Z 1.souboru: odstran_neslova,  Pocet klicovych slov: 4
Z 2.souboru: odstran,  Pocet klicovych slov: 4
Podobnost: 99.00%

Z 1.souboru: tiskni,  Pocet klicovych slov: 2
Z 2.souboru: tisk,  Pocet klicovych slov: 2
Podobnost: 97.33%

Dle sekvence klicovych slov:
----------------------------
Z 1.souboru: odstran_neslova,  Pocet klicovych slov: 3
Z 2.souboru: odstran,  Pocet klicovych slov: 3
Podobnost: 100.00%

Z 1.souboru: vrat_slovo,  Pocet klicovych slov: 3
Z 2.souboru: vrat,  Pocet klicovych slov: 3
Podobnost: 100.00%

Z 1.souboru: pridej_slovo,  Pocet klicovych slov: 4
Z 2.souboru: pridej,  Pocet klicovych slov: 4
Podobnost: 100.00%

Z 1.souboru: tiskni,  Pocet klicovych slov: 2
Z 2.souboru: tisk,  Pocet klicovych slov: 2
Podobnost: 100.00%

Z 1.souboru: main,  Pocet klicovych slov: 7
Z 2.souboru: main,  Pocet klicovych slov: 7
Podobnost: 100.00%
```

obr.15 – výstup programu při porovnání *podobných* souborů

Nyní byly přejmenovány a zpřeházeny globální proměnné i funkce, a navíc byly pozměněny i struktury těl funkcí:

```
---------------------------------
|  Pocet globalnich promennych  |
---------------------------------
V 1.souboru: 7
V 2.souboru: 5
---------------------------------
Shodnych dvojic: 16.67%
Typova podobnost: 33.33%

---------------------------------
|         Pocet funkci          |
---------------------------------
V 1.souboru: 3
V 2.souboru: 3
---------------------------------
| Nejpodobnejsi dvojice funkci  |
---------------------------------
```

```
Dle sekvence slov:
------------------
Z 1.souboru: main,  Pocet klicovych slov: 11
Z 2.souboru: main,  Pocet klicovych slov: 9
Podobnost: 86.21%

Z 1.souboru: cti_radek,  Pocet klicovych slov: 4
Z 2.souboru: cti_radek,  Pocet klicovych slov: 10
Podobnost: 56.25%

Z 1.souboru: chyba,  Pocet klicovych slov: 7
Z 2.souboru: chyba,  Pocet klicovych slov: 3
Podobnost: 30.61%


Dle sekvence klicovych slov:
----------------------------
Z 1.souboru: main,  Pocet klicovych slov: 11
Z 2.souboru: main,  Pocet klicovych slov: 9
Podobnost: 72.73%

Z 1.souboru: cti_radek,  Pocet klicovych slov: 4
Z 2.souboru: cti_radek,  Pocet klicovych slov: 10
Podobnost: 30.00%

Z 1.souboru: chyba,  Pocet klicovych slov: 7
Z 2.souboru: chyba,  Pocet klicovych slov: 3
Podobnost: 28.57%
```

obr.16 **–** výstup programu při porovnání souborů *na hraně rozpoznatelnosti*

Pro *odlišné* soubory je výstup programu následující:

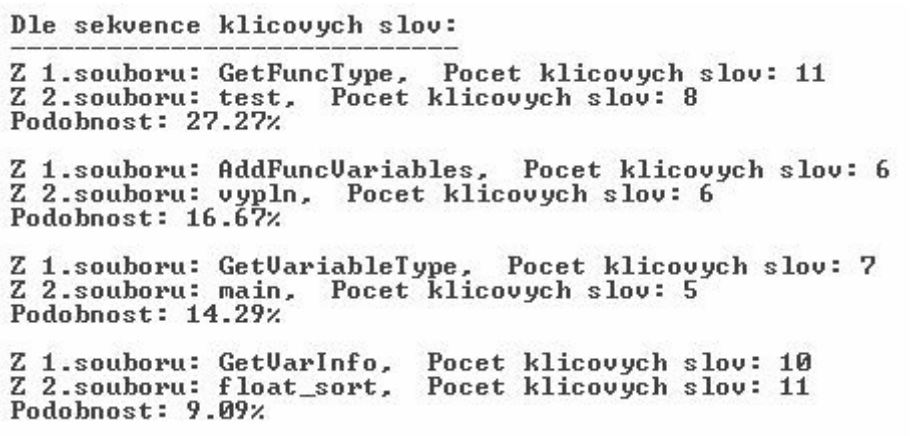
```
---------------------------------
¦  Pocet globalnich promennych  ¦
---------------------------------
V 1.souboru: 25
V 2.souboru: 0
---------------------------------
Shodnych dvojic: 0.00%
Typova podobnost: 0.00%

---------------------------------
¦          Pocet funkci         ¦
---------------------------------
V 1.souboru: 15
V 2.souboru: 4
---------------------------------
¦ Nejpodobnejsi dvojice funkci  ¦
---------------------------------
Dle sekvence slov:
------------------
Z 1.souboru: GetVarInfo,  Pocet klicovych slov: 11
Z 2.souboru: test,  Pocet klicovych slov: 8
Podobnost: 5.77%

Z 1.souboru: CheckTypes,  Pocet klicovych slov: 6
Z 2.souboru: main,  Pocet klicovych slov: 6
Podobnost: 5.26%

Z 1.souboru: GetFuncType,  Pocet klicovych slov: 7
Z 2.souboru: vypln,  Pocet klicovych slov: 5
Podobnost: 4.55%

Z 1.souboru: AddFuncVariables,  Pocet klicovych slov: 10
Z 2.souboru: float_sort,  Pocet klicovych slov: 11
Podobnost: 3.21%
```



obr.17 **–** výstup programu při porovnání *odlišných* souborů

4. Závěr

Cílem projektu bylo seznámení se s problematikou porovnávání textových souborů a jednorozměrných signálů a implementace algoritmu pro porovnání dvou programových souborů. Projekt může sloužit jako pomůcka při odhalování plagiátorství, avšak v některých situacích testy ukázaly, že výsledek u vyhodnocování podobnosti těl funkcí není dostatečně průkazný. To je pravděpodobně způsobeno neúplnou syntaktickou analýzou, která je zde využita jen u základních bodů. Možným vylepšením tak může být použití úplné syntaktické analýzy jazyka C, jež by mohla současně odhalovat i syntaktické chyby programu. Zavedení modulárního systému pro porovnávání zdrojových kódů jiných programovacích jazyků je další možností pro vylepšení.

Literatura

[1] Češka, M., Rábová, Z., Hruška, T., Máčel, M.: *Překladače*. SNTL, Praha, 1986.
[2] Uhlíř, J., Sovka, P.: *Číslicové zpracování signálů*. ČVUT, Praha, 2002.