Dokumentace úlohy CLS: C++ Classes v PHP 5 do IPP 2015/2016
Jméno a příjmení: Klára Nečasová
Login: xnecas24

Úvod
Cílem projektu bylo vytvořit skript `cls.php` v jazyce PHP 5, který provádí analýzu dědičnosti mezi třídami v hlavičkovém souboru jazyka C++11. Při řešení bylo využito objektově orientovaného přístupu pro snazší manipulaci s informacemi o jednotlivých třídách.
Činnost výše zmíněného skriptu lze rozdělit do následujících částí:

- zpracování parametrů příkazové řádky,
- zpracování vstupu a uložení informací do objektů tříd,
- zajištění dědičnosti,
- generování požadovaných výstupů.

Zpracování parametrů příkazové řádky
Užitečné informace o vstupních parametrech skriptu jsou uloženy ve třídě `Params`, kde je uložen název vstupního či výstupního souboru, případně další uživatelem specifikované parametry se svými hodnotami.

Zpracování vstupu a uložení informací do objektů tříd
Ke zpracování vstupu slouží funkce `parseInputFile()`. Samotné zpracování je realizováno pomocí konečného automatu, v průběhu zpracování jsou vytvářeny příslušné objekty, které reprezentují specifické části vstupního souboru. Třída `TFile` reprezentuje celý vstupní soubor. Obsahuje pole objektů třídy `TClass`, které reprezentuje jednotlivé třídy jazyka C++. Každá třída obsahuje atributy a metody. Proto bylo vytvořeno pole objektů třídy `TMethod` , které obsahuje pole objektů třídy `TArgs`, a pole objektů třídy `TAttribute`.

Zajištění dědičnosti
Každá třída si uchovává údaje o třídách, od kterých dědí. Tyto informace jsou uloženy v poli `parentsClasses`. Jedná se o asociativní pole (`nazevRodice`, `modifikatorPristupu`). Tyto informace jsou využity při generování výstupního XML souboru. Kromě toho jsou uchovávány také informace o třídách, které od dané třídy dědí (pole `childrens`), a to z důvodu využití těchto informací při generování stromu dědičnosti mezi třídami. Pokud nějaká třída dědí od jedné či více tříd, jsou z rodičovských tříd zkopírovány metody i atributy (metoda `copyClass()`). Kontrolu redefinice atributu či metody provádí metody `attrRedefinition()` a `methodRedefinition()`. Detekci konfliktů, ke kterým může dojít, pokud třída dědí od dvou a více tříd, zajišťuje metoda `checkConflict()`. Konfliktu je možné předejít použitím direktivy `using`, která zpřístupňuje metodu či atribut v daném jmenném prostoru, což zajišťuje metoda `addUsing()`. Všechny uvedené metody jsou metody třídy `TClass`.

Generování požadovaných výstupů
Pro generování požadovaného XML výstupu byla využita knihovna `XMLWriter`.
Pro vygenerování stromu dědičnosti mezi třídami byla použita funkce `generateInheritanceTreeXML()`, která za pomocí rekurzivního volání pomocné funkce `generateInheritanceTree()` prohledá objekty zpracovávaných tříd. Důležitým krokem bylo správné rozhodnutí o tom, zda se jedná o abstraktní či konkrétní třídu.
Pokud byl zadán přepínač `--details` bez specifikace konkrétní třídy, potom funkce `generateDetailsXML()` zajistí vygenerování popisu všech členů tříd. V případě, že byl přepínač `--details` zadán společně se specifikací konkrétní třídy (`--details=class`), zavolá se stejná funkce a zpracují se informace pouze o požadované třídě s názvem `class`. V této funkci bylo třeba zajistit potlačení výpisu těch elementů, které již neobsahují další informace, a také potlačit výpis privátních členů zděděných tříd.
Pokud byl zadán přepínač `--search=XPATH`, je v dosavadně vygenerovaném výstupu provedeno vyhledání příslušných elementů dle XPATH výrazu a teprve poté je výsledek naformátován a vypsán na výstup.