Dokumentace úlohy SYN: Zvýraznění syntaxe v Perl do IPP 2011/2012
Jméno a příjmení: Tomáš Válek
Login: xvalek02

Popis implementace od spuštění skriptu až po ukončení:
Po uvedených konstantách a globálních proměnných se spouští první podprogram.
**Podprogram params()** Skript po spuštění zpracuje parametry příkazového řádku. Ošetří případné chyby jako vícekrát zadaný parametr nebo parametr –help zadaný současně s nějakým jiným parametrem. Je-li zadán parametr –help, zavolá se podprogram help() pro výpis nápovědy a skript končí. Pokud parametr/y nevyhovuje, zavolá podprogram print_error_and_exit() s návratovým kódem 1 ke které se dostanu později.

Po úspěšném zpracování parametrů voláme:
**Podprogram get_content()**, který otevře soubory zadané uživatelem, zkopíruje je do daných proměnných a soubory uzavře. Zároveň kontrolujeme zda se nám soubory povedlo otevřít, pokud ne zavoláme podprogram print_error_and_exit() s návratovým kódem 2. Pokud nebyl zadán název vstupního souboru, použije se STDIN.

Nastává cyklus, který pro <u>každý řádek</u> formátovacího souboru převede regulární výraz(dále RV) na perlovský, zvýrazní syntaxi a uloží pozice tagů do polí. Tedy get_perl_reg_expr() a syntax() se volají v cyklu pro každý řádek.
**Podprogram get_perl_reg_expr()** Oddělíme RV od seznamu formátovacích parametrů. Zkontrolujeme zda je RV validní, zde uvedu pouze stručný náznak více je v kódu. Např. se nesmí vyskytovat jiná kombinace za znakem `%` než: `sadlLwWtn.|!*+()%` Dále znaky: `.+*|!` se nesmí vyskytovat dvakrát a více za sebou. Znak `%` se nesmí vyskytovat třikrát a více. RV nesmí začínat znaky `.|*+` a končit znaky `.|!(` Za znakem + se nesmí vyskytovat např. `*` a za znakem + nesmí být `*` atp. Převedeme RV zadaný uživatelem na perlovský RV a zkontrolujeme pomocí eval(`eval {/$perl_expr/;};`), zda uživatel zadal správný RV. Pokud při převodu RV nastala chyba voláme podprogram print_error_and_exit() s návratovým kódem 4.

Nyní máme validní RV a můžeme si označit syntaxi.
**Podprogram syntax()** Oddělíme formátovací parametry od RV a zavoláme

**podprogram check_valid_html_format(),** který nám zkontroluje zda jsou formátovací parametry(dále FP) správně zadány. Např. zda číslo v HEX u color je 6-ti ciferné a obsahuje pouze znaky [0-9A-F]. U velikosti fontu se zaměřujeme na velikost, která je dána od 1 do 7 včetně. Dále zda jsou FP ve správném formátu podle zadání. Pokud FP nevyhoví, tzn. není validní volá se podprogram print_error_and_exit() s návratovým kódem 4. Podle počtu FP na řádku je zavolán

**podprogram push_tag_to_array()** Vytváří dvě pole tagů. Jedno pole obsahuje počáteční tagy (, ,..) a druhé pole jsou koncové tagy(, ,..). Pole je ve tvaru:index_tagu#tag# např. 1##. Do pomocné proměnné se označí syntaxe. Vyhledají se počáteční tagy. Ty se uloží se do pole počátečních tagů s indexem na kterém se tag nacházel a za něj se uloží o jaký tag se jedná. Analogicky se provede vyhledání koncových tagů a uložení do pole koncových tagů. Indexy musí odpovídat vstupnímu textu. S tím si musíme poradit. Tzn. že si musíme pamatovat počet nalezených tagů, délku tagu a dále pokud byl zadán tag který označil prázdný řetězec. Tag který označil prázdný řetězec se do pole tagů nezahrnuje(prázdné řetězce neoznačujeme). Pro názornost uvedu krátký příklad:
vstup:Dobry den
formátovací soubor:
Dobry den italic
den bold

1 průchod cyklem v podprogramu **syntax()**
RV označil: Dobry den
Stav pole počátečních tagů:0## a pole koncových tagů:9##

2 průchod cyklem v podprogramu **syntax()**
RV označil:Dobry den
Stav pole počátečních tagů:0##6## a pole koncových tagů:9##9##

Máme přesné indexy počátečních a koncových tagů, zbývá označit text HTML tagy a uložit v případě potřeby uživatele do souboru. K označení textu slouží
**Podprogram print_text()** Označí text. V cyklu projdeme indexy vstupního textu od konce a pokud se index shoduje s indexem v poli koncových tagů, tak jej vložíme. Pole koncových tagů procházíme také od konce, abychom zaručili správné pořadí vložení. Po té procházíme pole počátečních tagů a pokud se indexy shodují vkládáme počáteční tag. V poli tagů tagy najdeme na pozici:pozice v poli tagů+2, protože 0## a  je na 2 pozici v poli. Takto procházíme obě pole tagů a vkládáme od konce vstupního textu pomocí funkce substr. Výsledný označený text máme v proměnné $file_output. Pokud byl aktivován parametr –br, tak před konec řádku vložíme tag <br \/>.

Kompletně označený text máme v proměnné a už nám zbývá provést výpis.
**Podprogram print_out()** Provede výpis textu na STDOUT pokud nebyl zadán parametr –output a nebo do souboru. Pokud se nepovede práce se souborem volá se podprogram print_error_and_exit() s návratovým kódem 3.
Program je u konce a končí s návratovou hodnotou 0.

Pomocné podprogramy:
**print_error_and_exit()** Vypisuje na STDERR chybu, která nastala a ukončí skript s odpovídajícím návratovým kódem. Parametry tohoto podprogramu jsou:číslo chyby a jméno souboru(pokud se jedná o soubor) v kterém chyba nastala.
**help()** Výpis nápovědy.

**Rozšíření NQS:** Implementováno v podprogramu get_perl_reg_expr() viz. výše.