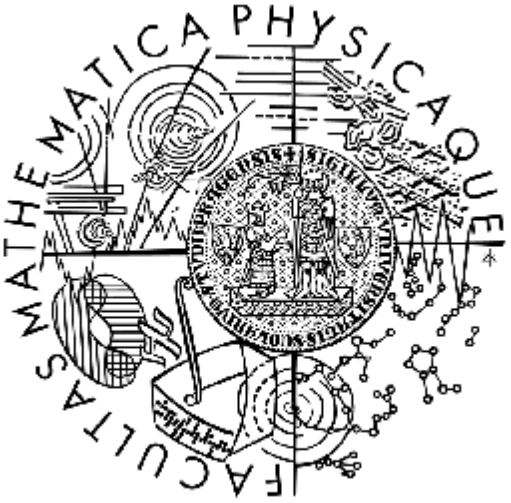
Univerzita Karlova v Praze
Matematicko-fyzikální fakulta

BAKALÁŘSKÁ PRÁCE



Pavol Gajarský

Textový editor se syntaxí řízenými funkcemi

Katedra softwarového inženýrství

Vedoucí bakalářské práce: RNDr. David Bednárek

Studijní program: Informatika, programování

2008

Ďakujem pánovi RNDr. Davidovi Bednárekovi za odborné vedenie, za ochotu a čas, ktorý mi venoval počas písania tejto bakalárskej práce.

Prehlasujem, že som svoju bakalársku prácu napísal samostatne a výhradne s použitím citovaných zdrojov. Súhlasím so zapožičiavaním práce.

V Prahe dňa 4.8.2008 Pavol Gajarský

Obsah

Názov práce: Textový editor se syntaxí řízenými funkcemi
Autor: Pavol Gajarský
Katedra: Katedra softwarového inženýrství
Vedúci bakalárskej práce: RNDr. David Bednárek
E-mail vedoúceho: David.Bednarek@mff.cuni.cz
Abstrakt:
Navrhnout a implementovat textový editor programátorského stylu s funkcemi řízenými syntaxí editovaného jazyka. Vhodným způsobem zadaná syntaktická pravidla budou sloužit jednak k vizuálnímu odlišení syntaktických konstrukcí, jednak k syntaxí řízeným funkcím jako průběžná indentace, formátování bloku, doplňování uzavíracích závorek, klíčových slov či XML elementů.
Kľúčové slová: editor, zvýrazňovanie syntaxie, indentácia

Title: Text editor with syntax-directed functions
Author: Pavol Gajarský
Department: Department of Software Engineering
Supervisor: RNDr. David Bednárek
Supervisor's e-mail address: David.Bednarek@mff.cuni.cz
Abstract:
Design and implement a programmer's text editor with syntax directed functions. Syntax rules submitted in a suitable format shall be used to visually distinguish syntactic constructs and to drive various functionality like automatic indentation, formatting, and auto-completion of parentheses, keywords, or XML elements.
Keywords: editor, syntax highlighting, indentation

1 Úvod

Zdrojový kód ľubovoľného programovacieho jazyka nie je len obyčajnou postupnosťou znakov. Ide o text, ktorý má charakteristickú štruktúru. Tá je v jazykoch s blokovou štruktúrou vyjadrená napríklad definíciami funkcií a procedúr, časťami s deklaráciami premenných alebo vetveniami. V jazykoch odvodených od XML zasa jednotlivými tagmi. Jedná sa nie len o štruktúru určenú syntaxiou daného jazyka ale taktiež aj o logickú štruktúru programu. Jej vyjadrenie môžeme dosiahnuť práve vhodne zvoleným zápisom zdrojového kódu. Preto použitie formátovacích metód pre obyčajný text ako usporiadanie textu do odstavcov, zoznamov alebo zarovnanie na stred obrazovky je pre zdrojový kód nevhodné. Užívateľovi je nutné poskytnúť zobrazenie programu vo forme v ktorej je jednoduché sa orientovať bez zdĺhavého analyzovania. Pomocou neho by mal byť schopný na prvý pohľad odlíšiť jednotlivé syntaktické konštrukcie, to či je text súčasťou komentára, dané slovo identifikátorom alebo kľúčovým slovom jazyka a podobne. Okrem vizuálneho spracovania textu je dôležité si uvedomiť že operácie používané pre obyčajný text je nutné implementovať pre zdrojový kód rozdielne resp. použitie niektorých nemá zmysel ako vkladanie tabuliek, obrázkov.

1.1 Prečo implementovať programátorský editor

Z vyššie spomenutých rozdielov medzi editovaním obyčajného textu a zdrojového kódu je zrejmé že použitie klasických nástrojov je pre programátora nevhodné a ťažkopádne. Zdrojový kód každého programovacieho jazyka má svoju typickú štruktúru, ktorá je daná jeho syntaxiou. Z jej znalosti je možné vybudovať editor, ktorý využije všetky jej črty k tomu aby poskytol užívateľovi prostriedky na čo najjednoduchšiu editáciu. Netýka sa to len zvýrazňovania syntaxie, čo je bez pochyby najväčšou vizuálnou pomocou, ale editor môže takisto poskytovať funkcie, ktoré zvýšia efektivitu práce programátora. Takouto funkciou je napr. doplňovanie zátvoriek, vkladanie častých syntaktických konštrukcií akými sú zápisy cyklov a vetvení prostredníctvom šablón ale takisto aj riadenie indentácie. Pri nej sa editor snaží vystihnúť príslušnosť jednotlivých príkazov k nadriadeným blokom vhodným vkladaním bielych znakov na začiatky riadkov. Takýto editor sa stáva pre programátora neoceniteľným pomocníkom pri práci, pretože mu poskytuje prostredie v ktorom sa môže plne sústrediť na písanie kódu, teda na to, čo má byť jeho jedinou úlohou. Nie je zaťažovaný detailmi ako formátovanie textu, to zaňho urobí samotný editor.

1.2 Cieľ tejto práce

V dnešnej dobe existuje množstvo editorov daného zamerania od tých najjednoduchších, ktoré podporujú iba zvýrazňovanie syntaxie (ang. syntax highlighting) až po sofistikované nástroje, vyvíjané a zlepšované desiatky rokov, ktoré podporujú väčšinu z dnes známych funkcií. V mojich silách samozrejme nie je týmto nástrojom konkurovať. Budem sa snažiť implementovať rozumnú a zaujímavú podmnožinu funkcií typických pre editory tejto kategórie a pridať niečo navyše.

V práci som sa zameral hlavne na nasledujúce problémy :

1. Podporované jazyky

Jedným z problémom u väčšiny editorov je podpora len niekoľkých preddefinovaných programovacích jazykov. Ak potrebuje užívateľ programovať v jazyku, ktorý nie je podporovaný daným editorom musí sa obrátiť inde minimálne dovtedy pokým jeho podpora nebude pridaná v ďalších verziách. To je samozrejme nevyhnutné pri jazykoch, ktoré si vyžadujú špecifický prístup ale u klasických jazykoch s blokovou štruktúrou je zavedenie takejto podpory možné. V danej práci som zvolil prístup pomocou vytvárania tzv. štýlov. Štýl predstavuje pre daný programovací jazyk akúsi obálku nesúcu všetky nastavania, ktoré sú potrebné pre zaistenie vykonávania funkcií ako syntax highlighting, doplňovanie zátvoriek, kľúčových slov, XML elemetov (ak je jazyk podmnožinou XML) a riadenie indentácie.

2. Indentácia

Druhý z problémov sa dotýka indentácie. Väčšina dnešných editorov podporuje jednu z dvoch foriem. A to buď blokovú alebo tzv. smart (inteligentnú) indentáciu. Pri blokovej indentácii dochádza k odsadzovaniu riadkov vnútri bloku o istý počet bielych znakov (tabulátorov alebo medzier) od jeho začiatku. Týmto spôsobom sa zvýrazňuje štruktúra zdrojového kódu. Na prvý pohľad je potom možné rozoznať, ktoré časti kódu sú podriadené resp. sú súčasťou nejakej štruktúry akou je cyklus alebo vetvenie. Pri smart indentácii záleží jednak na programovacom jazyku a jednak na použitom editore, pretože každý si ju môže interpretovať svojím spôsobom. V princípe ale ide o kombináciu blokovej indentácie a niektorých indentačných postupov overených praxou ako odsadzovanie v rámci zátvoriek, zarovnanie else na úroveň najbližšieho nespárovaného if, koniec bloku na úroveň jeho začiatku a podobne. Popisované druhy indentácií sú v editoroch zabudované na pevno a teda nie je možné ich prispôsobiť svojim potrebám. Niektoré editory umožňujú do určitej miery meniť ich nastavenie napr. prepínaním medzi nimi ak ich majú implementované obe, alebo zakázaním niektorých typov indentácií typicky ide o odsadzovanie zátvoriek alebo vetvení if-else, ale neumožňujú ich celkovú zmenu.

Spôsobov indentácie je mnoho, každému užívateľovi (programátorovi) vyhovuje niečo iné. V nasledujúcej tabuľke sú uvedené štyri najpoužívanejšie spôsoby indentácie v rámci bloku (na ukážke je použitý jazyk C++)

if (*podmienka*)**{** *príkaz 1;* *príkaz 2;* *…* **}**	**if** (*podmienka*) **{** *príkaz 1;* *príkaz 2;* *…* **}**
if (*podmienka*) **{** *príkaz 1;* *príkaz 2;* *…* **}**	**if** (*podmienka*) **{** *príkaz 1;* *príkaz 2;* *…* **}**

Väčšina zo súčasných nástrojov používa práve jeden z nich a tým núti užívateľa k prispôsobeniu sa. Programátor si teda nemôže s prechodom do iného prostredia preniesť aj svoje zvyklosti. Tento problém som sa rozhodol vyriešiť prostredníctvom zavedenia indentačných pravidiel, definovaných užívateľom pre každý programovací

jazyk. Pomocou nich bude vykonávaná indentácia v celom dokumente. Programátorovi pri zmene svojich formátovacích návykov stačí uplatniť príslušné zmeny v pravidlách.

Cieľom tejto práce je teda navrhnúť editor programátorského štýlu, ktorý by zabezpečoval funkcie ako zvýrazňovanie syntaxie, doplňovanie zátvoriek, kľúčových slov a XML elementov a riadil indentáciu pomocou užívateľom zadaných pravidiel. Editor nebude podporovať vopred zvolenú množinu programovacích jazykov, ale poskytne užívateľovi prostriedky, pomocou ktorých ich bude môcť do editoru pridať. To sa v editore bude riešiť pomocou vytvárania štýlov. Každému programovaciemu jazyku bude priradený štýl, ktorý vytvára akúsi obálku pre všetky nastavenia, ktoré sú nutné pre zabezpečenie spomínaných funkcií. Editor bude založený výlučne na používaní štýlov. Teda na každom editovanom dokumente bude uplatnený niektorý z nich. Práve z tohto dôvodu bude v editore vytvorený preddefinovaný prázdny štýl, ktorý je možné použiť na editovanie obyčajného textu.

Editor bude ďalej podporovať obvyklé funkcie dostupné z hlavného menu alebo panela nástrojov. A to :

- otváranie a ukladanie súborov
- tlačenie, poskytnutie náhľadu pre tlačením
- zoznam posledných 4 otvorených dokumentov
- neobmedzené undo
- kopírovanie, vystrihnutie a vkladanie textu
- vyhľadávanie a nahradzovanie textu
- editovanie viacerých dokumentov súčasne
- rozloženie okien na ploche a prepínanie sa medzi nimi
- zobrazenie informácii o práve editovanom dokumente v stavovom riadku

1.3 Štruktúra textu

Text tejto bakalárskej práce má nasledujúcu štruktúru. V druhej kapitole je rozvedená problematika súvisiaca s implementáciou editora programátorského štýlu. V jej úvode je načrtnutý problém z teoretickej časti, v podkapitole *Ako implementovať programátorský editor* sa venujem možnostiam implementácie a spôsobu, ktorý som nakoniec zvolil. Nasledujúca kapitola 3 zoznamuje užívateľa so samotným programom, jeho ovládaním a popisom prostredia. Nasleduje kapitola s názvom *Vývojová dokumentácia*, ktorá sa snaží priblížiť problémy vzniknuté pri implementácii a ich riešenie. V kapitole 5 je obsiahnuté porovnanie s existujúcimi programami rovnakého alebo podobného zamerania. Nasledujúca kapitola 6 obsahuje popis priloženého CD obsahujúceho výsledný editor. Nakoniec v kapitole 7 *Záver* hodnotím dosiahnutie vytýčených cieľov a uvádzam možnosti o ktoré by táto práca mohla byť v budúcnosti rozšírená. V samom závere je pripojený zoznam literatúry z ktorej som čerpal pri písaní tejto bakalárskej práce.

2 Súvisiaca problematika

Pri spracovaní textu akým je zdrojový kód programovacieho jazyka sa používajú poznatky z teórií prekladačov, automatov a gramatík. Tie budú uplatnené aj v tomto editore a to konkrétne poznatky k konštrukcie konečných automatov a lexikálnej analýzy.

Pri lexikálnej analýze nastáva rozklad vstupného textu na postupnosť lexikálnych elementov akými sú napr. kľúčové slova, identifikátory alebo čísla. Dochádza k postupnému načítavaniu znakov zo vstupu a zostavovaniu lexikálnych symbolov, z ktorých každý predstavuje logicky súvisiacu postupnosť znakov. Táto postupnosť sa označuje ako lexém. Každému rozoznanému symbolu alebo elementu sa priraďuje jednoznačný identifikátor, ktorý sa nazýva token (zvyčajne je reprezentovaný číslom). Niekedy token obsahuje atribút ktorý ho bližšie určuje napr. každý rozoznaný reťazec produkuje rovnaký token, ale ten nehovorí nič o jeho obsahu, v tomto prípade bude na jeho predanie použitý spomínaný atribút, ďalším príkladom môže byť situácia kedy jeden token zastupuje celú množinu operátorov, potom atribút slúži na presne určenie o ktorý z nich sa jedná. V tomto editore si vystačíme s tokenmi bez atribútov, keďže pri zvýrazňovaní literátov akými sú reťazce alebo číslo nepotrebujeme poznať ich obsah a každému rozoznávanému elementu bude priradený jedinečný token. Pri lexikálnej analýze sa zvyčajne vynechávajú biele znaky, tak tomu bude aj v tomto editore.

Pre dostatočne rýchlu lexikálnu analýzu sú iba dve možnosti implementácie analyzátoru a to :

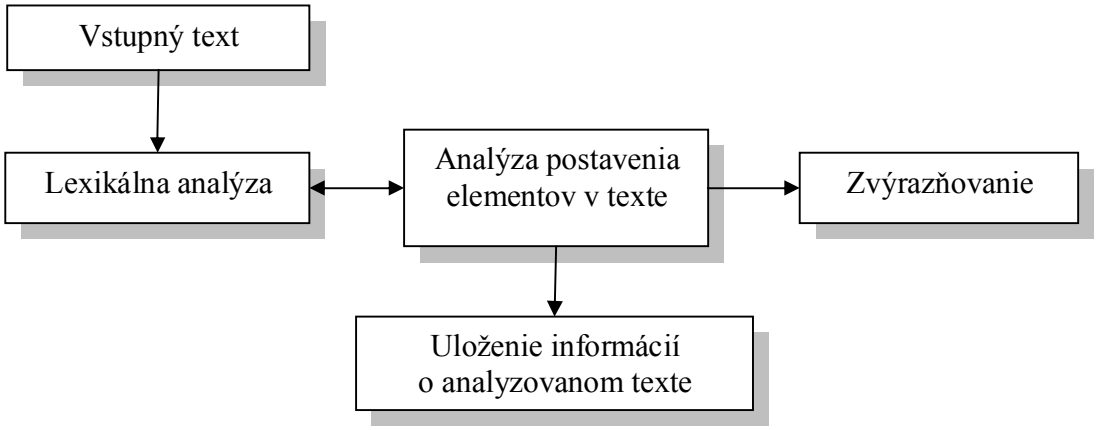
- Priama implementácia v danom programovacom jazyku
- Implementácia pomocou konečného automatu

V prvom prípade by bolo nutné vytvoriť lexikálny analyzátor osobitne pre každý jazyk, ktorý má byť podporovaný. Takýto postup je bežný v mnohých editoroch. Realizuje sa napr. vytvoreným osobitnej triedy pre každý z programovacích jazykov, ktorá zabezpečuje jeho lexikálnu analýzu a poskytuje jednotný interface pre zvyšné časti editora. Týmto interfaceom je typicky spôsob predávania tokenov. Samotná analýza je skrytá vo vnútri triedy a pre každý programovací jazyk môže byť implementovaná úplne inak. Tento spôsob je najnáročnejší na implementáciu, ale zároveň prináša najlepšie výsledky, keďže analyzátor je možné čo najviac prispôsobiť syntaktickým vlastnostiam daného jazyka. Jeho nevýhoda je v tom, že pridanie podpory pre ďalší programovací jazyk nemôže byť urobené bez zásahu programátora. Preto pre potreby tejto bakalárskej práce, v ktorej užívateľ sám rozhodne o tom ktorý jazyk bude editor spracovávať je daný spôsob nevhodný. Rozhodol som sa preto pre druhý spôsob, ktorý prináša síce o trochu horšie ale porovnateľné výsledky. Hlavnou výhodou je však jeho univerzálnosť, keďže konečný automat realizujúci lexikálnu analýzu je možné vytvoriť rovnakým postupom pre každý jazyk zo znalosti jeho elementov, ktoré má rozpoznávať.

Konečný automat je teoretický výpočetný model. Pozostáva z konečného počtu stavov, ktoré odrážajú priebeh výpočtu. Automat sa v každom okamihu nachádza v práve jednom z nich. Na začiatku je vo svojom počiatočnom stave. Po každom prečítanom znaku zo vstupu prejde do stavu určeného prechodovou funkciou (tá je zvyčajne realizovaná prechodovou tabuľkou). V prípade, že sa dostane do niektorého z prijímacích stavov došlo k rozpoznaniu elementu daného jazyka, automat sa

reštartuje, teda vráti do svojho počiatočného stavu a pokračuje dovtedy pokiaľ nie je prečítaný celý vstup. V našom prípade bude konečný automat realizovaný nad abecedou pozostávajúcou z tlačiteľných znakov ascii tabuľky.

Časť editor zabezpečujúca zvýrazňovanie má nasledovnú štruktúru.



Lexikálna analýza vstupného textu je realizovaná konečným automatom. Po rozpoznaní nejakého elementu je predaný jeho token na analýzu postavenia v texte, v ktorej sa určí či je dané slovo reprezentované tokenom možné zobraziť. Následné môže byť slovo predané do častí starajúcich sa o zvýrazňovanie a ukladanie informácií vyskytujúcich sa v texte. Napr. pri otváraní nového dokumentu je analyzovaný celý text a musia byť uložené informácie o každom riadku, ale zvýrazniť stačí iba jeho viditeľnú časť. Následne po zoscrollovaní na nezvýraznenú časť textu stačí iba jej sformátovanie, už nie je nutné ukladať informácie o texte, pretože tie už boli spracované na začiatku.

Vo fáze lexikálnej analýzy kompilátora sa spracováva celý vstupný text. Avšak pri zvýrazňovaní syntaxie sa to deje iba na začiatku pri otváraní dokumentu. Pri malých zmenách ako vloženie znaku nie je možná analýza celého textu, pretože by to nesmierne spomalilo celý editačný proces. V tomto prípade dôjde k analýze iba riadku na ktorom sa daná zmena uskutočnila. Ale tu nastáva problém v akom stave sa má ocitnúť automat na začiatku analýzy. Ak je riadok súčasťou viacriadkového komentára, tak nemôžeme uviesť automat do jeho počiatočného stavu. Podobne je to pri jazykoch odvodených z XML. Pri nich je dôležité zistiť, či sa editovaný riadok nachádza v rámci nejakého otvoreného XML elementu na vyšších riadkoch. Aj kvôli tomuto prípadu sa v editore pre každý riadok udržiava stav výskytu viacriadkových komentárov a otvorenosti XML tagov. Z týchto stavov sa pri analyzovaní textu najprv určí v akom kontexte sa nachádza jeho začiatok. Ďalším typickým problémom je neukončený reťazec na konci riadka alebo načítanie neznámeho znaku (napr. znaku k diakritikou). Je nutné sa z týchto chýb úspešne zotaviť a pokračovať ďalej. V prípade kompilátora by došlo k ukončeniu procesu a nahláseniu príslušných chýb. V tomto prípade to možné nie je. Editor musí zabezpečiť korektné zvýrazňovanie i v prípade syntakticky chybného vstupu. Častým riešením ako sa zotaviť z chýb je doplnenie chýbajúceho znaku. Toto riešenie sa používa aj v tomto editore pri neukončenom reťazci, pri načítaní neznámeho znaku dôjde k jeho ignorovaniu a text ktorého je súčasťou bude sformátovaný ako obyčajný text, automat sa uvedie do počiatočného stavu a analýza pokračuje ďalej.

Samostatnou kapitolou je rýchlosť, pretože práve tá je príčinou kompromisov medzi zamýšľanou funkcionalitou, ktorú má editor poskytovať a výsledným riešením. Tento problém je bližšie spracovaný vo *Vývojovej dokumentácii*.

2.1 Ako implementovať programátorský editor

Keďže ide svojím spôsobom o textový editor je nutné mať na jeho implementáciu prostredie, ktoré je schopné spracovávať viacriadkový text. Taktiež by malo poskytovať rozhranie pomocou ktorého je možné zistiť aktuálnu pozíciu kurzora, viditeľný výrez obrazovky, kvôli tomu, že pre rýchle spracovanie dokumentu je možné formátovať iba jeho viditeľnú časť. Umožniť programové vkladanie textu, kvôli zmene indentácie, vkladaniu šablón a zabezpečeniu operácií ako vloženie textu zo schránky (paste) a jeho vystrihnutie (cut). V neposlednej rade musí umožniť uplatnenie zvoleného formátovania na ľubovoľnú časť textu kvôli implementácii zvýrazňovania syntaxie (syntax highlightingu). Formátovanie by malo pozostávať z nastavenia farby písma, farby jeho pozadia, rezu, veľkosti a typu. Minimum by sa dalo zhrnúť do nastavenia farby napr. ak má byť editor implementovaný v prostredí príkazového riadka.

Pre tento účel je možné použiť buď niektorú z existujúcich štandardných komponent, vytvoriť si vlastnú alebo prípadne použiť nejakú knižnicu na to určenú. V mojom prípade som sa rozhodol pre použitie štandardnej MFC komponenty CRichEditView. Tá zabezpečuje spomínané funkcie pomocou svojho interného formátu RTF (Rich Text Format) v ktorom si uchováva formátovanie textu. Ide o free document file format, teda formát, ktorého špecifikácia je voľne dostupná a na jeho použitie nie sú kladené žiadne obmedzenia. RTF uplatňuje formátovanie pomocou control kódov (ang. control codes). Ide o sekvenciu ktorá začína spätným lomítkom \ a končí po najbližší biely znak, zloženú zátvorku alebo začiatok ďalšieho control kódu. Každý kód určuje isté formátovanie pre text, ktorý po ňom nasleduje. Výsledné formátovanie dostaneme skladaním príslušných control kódov, teda ich zápisom za sebou.

príklad

niektoré z najviac používaných control codes

\b – nasledujúci text bude zobrazený tučne **\i** – nasledujúci text bude zobrazený kurzívou **\ul** – nasledujúci text bude podčiarknutý **\f**<N> – nasledujúci text bude zobrazený N-tým fontom z tabuľky písem **\fs**<N> – nasledujúci text bude mať veľkosť písma zadanú parametrom N

Ak chceme uplatniť dané formátovanie len na časť textu tak ho musíme uzavrieť do zložených zátvoriek

príklad

zápis v RTF formáte

{\i tento text bude napísaný kurzívou}{\b tento tučne}

výsledok vyzerá nasledovne

tento text bude napísaný kurzívou **tento tučne**

Teda samotné zvýrazňovanie syntaxie na určitom texte prebieha jeho prepisom do RTF kódu, ktorý odzrkadľuje nadefinované zvýrazňovanie elementov jazyka.

príklad

```
if (riadok == GetRichEditCtrl().GetLineCount() - 1) return -1;
```

Takto vyzerá minimálny RTF kód, ktorý musí byť vložený na zvýraznenie predchádzajúceho riadka.

```
{\rtf1\ansi{\fonttbl{\f0\fnil Courier New;}} {\colortbl
;\red0\green0\blue255;\red0\green0\blue0;\red0\green100\blue200;\\re
d100\green100\blue100;\red200\green0\blue0;\red100\green0\blue100;}\
cf1\b\f0\fs22 if\b0\cf3 (\cf4 riadok \cf5 == \cf4
GetRichEditCtrl\cf3 ()\cf5 .\cf4 GetLineCount\cf3 () \cf5 - \cf6
1\cf3 ) \cf1\b return\b0  \cf5 -\cf6 1\cf3 ;}
```

Samotný RTF formát okrem požadovaného formátovania akým je nastavenie farby písma, farby pozadia, rezu, veľkosti a typu umožňuje napr. orámovanie časti textu, vkladanie obrázkov, tabuliek, označenie časti textu, ktorá má byť iba na čítanie, skrývanie textu a množstvo ďalších funkcií. Tie samozrejme nie sú potrebné pre implementáciu funkcií, ktoré má zabezpečovať daný editor, ale je to ukážkou komplexnosti tohto formátu. Funkcie ako orámovanie textu alebo jeho skrývanie by však mohli byť použité pri zvýrazňovaní alebo pri implementácii funkcií akou je code folding, teda skrývanie zdrojového kódu, ktorý je súčasťou jedného bloku napr. tela metódy alebo viacriadkového komentára. Snáď jedinou nevýhodou komplexnosti tohto formátu, je že použitá komponenta musí všetky uvedený funkcie poskytnuté daným formátom spĺňať i keď nie sú po nej vyžadované. A to vedie k zbytočnej réžii navyše, ktorá môže spomaliť spracovanie veľkých dokumentov.

Použitá komponenta taktiež umožňuje funkcie ako je vrátenie poslednej zmeny, vyhľadávanie a nahradzovanie textu, vkladanie textu vo formáte RTF, vkladanie obrázkov a iných objektov, ktoré majú podporu v tomto formáte. Všetky tieto funkcie sú navrhnuté na použitie pri editovaní obyčajného textu ale nie pre editovanie zdrojového kódu. Preto museli byť naprogramované odznova, alebo minimálne pozmenené. Vkladanie bolo umožnené iba ak ide o nesformátovaný text. Pre obrázky, tabuľky a ďalšie objekty je vkladanie zakázané, pretože v texte obsahujúcom zdrojový kód nemajú čo hľadať. Operácia undo musela byť nahradená. Kvôli tomu že pri vrátení poslednej úpravy je nutné obnoviť informácie na riadkoch ktorých sa dotýka a napr. ak bol v poslednej úprave zmazaný viacriadkový komentár, tak jej vrátením sa opäť vloží a je nutné preformátovať text ktorý v ňou bude obsiahnutý. Pri nahradzovaní textu je nutné opäť dávať pozor napr. už na spomínaný komentár ak ho text ktorým je nahradzované obsahuje a pod. Štruktúra zdrojového kódu si vyžaduje odlišný prístup k editácii ako obyčajný text.

3 Užívateľská dokumentácia

3.1 Štýl

Štýl je možné charakterizovať ako súbor konfigurácii pre určitý programovací jazyk, ktorý umožňuje jeho správne editovanie. Editor je postavený výlučne na používaní štýlov, teda nie je možné editovať obyčajný text (ang. plain text) v pravom slova zmysle, ale je ho možné editovať použitím predpripraveného štýlu, ktorý je prázdny (neobsahuje žiadne zvýraznenie, šablóny, ...), čím v konečnom dôsledku dosiahneme rovnaký efekt.

Štýl je tvorený :

- Základnými informáciami o programovacom jazyku
- Zvýrazňovaním syntaxie
- Šablónami
- Indentačnými pravidlami

Základné informácie o programovacom jazyku

Obsahujú informácie ako popis štýlu, prípony súborov, ktoré obsahujú zdrojový kód, program, farbu pozadia a počiatočný text.

Popis štýlu je krátky jednoriadkový popis, ktorý sa zobrazuje v niektorých dialógoch programu (napr. pri vytvorení nového dokumentu). Ak nadefinujeme viac štýlov pre jeden jazyk, je dobré im dať výstižný popis napr. z ktorého prostredia sú prebrané resp. čo ich najviac vystihuje.

Prípony súborov so zdrojovým kódom sú využívané pri ich otváraní. Ak má súbor príponu priradenú niektorému štýlu, tak sa naň automaticky uplatní daný štýl (napr. pre C++ cpp h, pre Pascal pas). Ak nie, použije sa preddefinovaný prázdny štýl používaný pre editovanie obyčajného textu.

Program je aplikácia, ktorej je možné predať editovaný zdrojový kód (napr. html dokument webovému prehliadaču alebo zdrojový kód v C++ kompilátoru).

Farba pozadia je farba, ktorá bude použitá na pozadí editovaného dokumentu (typicky biela, modrá alebo čierna).

Počiatočný text je text, ktorý sa vloží po vytvorení nového dokumentu v danom štýle. Mal by obsahovať základnú kostru programu.

Zvýrazňovanie syntaxie

Obsahuje nastavenie zvýrazňovania kľúčových slov ich rozdelením do skupín, ktoré sú nositeľmi formátovacích konfigurácií, ďalej formátovanie špeciálnych elementov ako komentáre, reťazce uzavreté úvodzovkami a apostrofmi, obyčajný text, čísla a identifikátory. V tejto časti je takisto možné nadefinovať zobrazenie zátvoriek po nájdení páru a v prípade jeho nenájdenia nastaviť automatické doplnenie. Nakoniec je možné určiť, či sa daný štýl bude uplatňovať na dokumentoch so štruktúrou XML,

pretože si to vyžaduje iný prístup k zvýrazňovaniu a to, či sa budú pri zvýrazňovaní rozlišovať veľké a malé písmená.

Šablóny

Obsahujú šablóny zadané užívateľom, ktoré sa ponúkajú na výber z kontextového menu pri napísaní začiatku niektorej z nich. Užívateľ má možnosť nadefinovať 2 typy šablón a to obyčajné a XML, ktoré je možné použiť iba v XML dokumentoch.

Indentačné pravidlá

Obsahujú definície pravidiel ktorých uplatňovaním sa dokument indentuje, teda vhodným vkladaním bielych znakov na začiatky riadkov sa snaží vystihnúť blokovú štruktúru kódu. Ide o tzv. kontextové pravidlá, ktoré sa uplatňujú na základe predchádzajúceho kontextu, teda tvaru zdrojového kódu.

Prepojenie častí

Jednotlivé časti štýlu nie sú medzi sebou izolované, ale naopak ovplyvňujú sa a to nasledovne. Kľúčové slová, ktoré sú definované v časti so zvýrazňovaním sú jediné, ktoré editor pri analýze textu konečným automatom rozpoznáva. Indentačné pravidlá, ktoré sa vo svojej definícii odkazujú na nejaké kľúčové slovo, odkazujú sa práve na slovo z časti so zvýrazňovaním. Takisto v časti so zvýrazňovaním syntaxie je možné určiť či sa bude štýl používať na dokumentoch typu XML. V prípade, že nie, tak všetky nadefinované XML šablóny nebude možné uplatniť.

Práca so štýlmi

Štýl je možné pridať, premenovať, odstrániť a editovať. Pri pridaní štýlu je nutné zvoliť jednoznačný názov. Pri odstránení štýlu sa odstránia všetky jeho súčasti popísané vyššie. Nie je možné odstrániť ani premenovať štýl s názvom ot, ktorý je východiskový štýl pre obyčajný text (plain text).

3.2 Zvýrazňovanie syntaxie

V angličtine označované ako syntax highlighting. Ide o definovanie formátovania písma ako farba, rez, typ pre jednotlivé elementy jazyka akými sú kľúčové slová, komentáre, reťazce a podobne. V takto ofarbenom zdrojovom kóde je užívateľ schopný sa jednoduchšie orientovať a na prvý pohľad odlíšiť jednotlivé syntaktické konštrukcie jazyka.

Definovanie zvýrazňovania

Môžeme rozdeliť do 2 častí a to zvýrazňovanie kľúčových slov a zvýrazňovanie špeciálnych elementov akými sú komentáre, reťazce, identifikátory, čísla a zátvorky.

Editor má podporu pre dokumenty typu XML (napr. jazyk HTML). Označením, že štýl bude používaný na dokumente typu XML sa predpokladá že dokument je členení do štruktúry, ktorá pozostáva z nepárových a párových tagov ohraničených <,> medzi ktorými je umiestnený text, ktorý sa neanalyzuje. Teda akékoľvek zvýrazňovanie je možné len v rámci tagov. Všetko mimo nich sa interpretuje ako obyčajný text.

Zvýrazňovanie kľúčových slov :

Základom je vytvorenie skupiny. Tá tvorí akýsi obal pre kľúčové slová, ktorý v sebe zahŕňa rôzne nastavenia. Tieto nastavenia sa automaticky prenášajú na slová, ktoré sú v nej obsiahnuté, teda samotné kľúčové slovo nenesie žiadnu informáciu a preto jeho existencia mimo skupiny nemá zmysel. Nastavenie zvýrazňovania pre dané kľúčové slovo sa dosiahne jeho pridaním do niektorej z existujúcich skupín. Slová je možné do skupín ľubovoľne pridávať a odoberať, s tým, že slovo sa môže vyskytovať najviac v jednej z nich.

Skupine je možné nastaviť :

- Formátovanie písma – farba, rez a typ písma.
- Postavenie texte – ak je skupina definovaná ako skupina oddeľovačov, tak slová v nej obsiahnuté budú zobrazené bez ohľadu na postavanie v texte. Samozrejme musia ležať mimo komentárov, reťazcov uzavretých úvodzovkami alebo apostrofmi. Ak skupina danú vlastnosť nemá tak jej kľúčové slová budú zobrazené iba v prípade, že sú z oboch strán obklopené buď 1. bielymi znakmi, 2. slovami, ktoré sú súčasťou niektorej zo skupín oddeľovačov alebo 3. komentármi, reťazcami uzavretými úvodzovkami alebo apostrofmi (a samozrejme nie sú ich súčasťou).

Skupinu je možné pridať, zmazať, premenovať. Pri pridávaní musí byť skupine priradený jednoznačný názov. Pri jej zmazaní sú automaticky zmazané i slová ktoré obsahuje.

Definovanie kľúčových slov je dôležité, pretože sú ďalej využívané pravidlami v indentácii. Ak slovo nie je definované v tejto časti, tak ho konečný automat nebude rozpoznávať a teda ani nemôže byť uložená informácia o jeho nájdení a preto nebude môcť byť uplatnené žiadne pravidlo, ktoré ho obsahuje vo svojej definícii. Ak je dané kľúčové slovo potrebné pre indentačné pravidlá, ale nie je žiadúce jeho zobrazovanie, tak sa musí umiestniť do skupiny, ktorá bude mať nadefinované formátovanie rovnaké ako obyčajný text.

Ak chceme nadefinovať zobrazenie pre case sensitive (záleží na veľkosti písmen) programovací jazyk tak musíme nastaviť rozlišovanie veľkých a malých písmen v danom štýle zaškrtnutím príslušnej voľby. Podobne pre case insensitive (nezáleží na veľkosti písmen).

Zvýrazňovanie špeciálnych elementov :

Týka sa jednoriadkového a viacriadkového komentára, reťazcov uzavretých úvodzovkami a apostrofmi, identifikátorov, čísel a zátvoriek.

Pri každom z týchto elementov sa zadáva formátovanie písma, ktorým sa daný element, zobrazí. V prípade komentárov a reťazcov uzavretých úvodzovkami alebo apostrofmi sa myslí taktiež zvýraznenie ich obsahu, nie len ich začiatok a koniec.

Pri jednoriadkovom komentári je nutné zadať reťazec ktorý ho uvádza, teda reťazec ktorý udáva jeho začiatok. Koniec je implicitne daný koncom riadka.
Pri viacriadkovom komentári sa zadáva reťazec udávajúci jeho začiatok a koniec.

U reťazcoch uzavretých úvodzovkami alebo apostrofmi je možné zakázať ich formátovanie (teda ich obsah sa bude formátovať). To je možné uplatniť v jazykoch,

ktoré majú podporu iba jednej formy zápisu reťazca. K apostrofu a úvodzovke sa zadáva taktiež escape sekvencia z oboch strán, tá slúži k rušeniu ich významu, teda escapovaný apostrof alebo úvodzovku ide použiť v reťazci bez toho aby došlo k jeho ukončeniu.

príklad

C++

`"reťazec v \" jazyku C++"` escape reťazec pre " je \

Pascal

`'reťazec v jazyku''' Pascal'` escape reťazec pre ' je rozdelený na ľavú časť (') a pravú časť (').

Za obyčajný text sa považuje text, ktorý je umiestnený mimo komentárov, reťazcov uzavretých úvodzovkami alebo apostrofmi a ktorý neprodukuje žiadny token, teda nie je rozpoznávaný automatom. Dôležité je, že jedine u tohto elementu je možné nastaviť veľkosť písma. Tá sa automaticky vzťahuje na všetky kľúčové slová a ostatné elementy.

Za identifikátor sa berie reťazec začínajúci písmenom po ktorom nasledujú písmená alebo číslice. To je základná kostra, ktoré je však v mnohých jazykoch rozšírená o ďalšie znaky. Preto je možné zadať zoznam znakov, ktoré sa ešte môžu v identifikátore vyskytovať. (napr. v C++ by to bol zoznam [_]). Formátovanie identifikátorov je možné vypnúť, potom sa zobrazujú rovnako ako obyčajný text.

U čísel sa zobrazovanie týka celých čísel v desiatkovej alebo šestnástkovej sústave a reálnych čísel s pevnou desatinnou časťou alebo zapísaných v semilogaritmickom tvare. Formátovanie čísel je možné vypnúť, potom sa zobrazujú rovnako ako obyčajný text.

Escape znak ruší význam znaku po ňom nesledovanom. Je možné zadať i reťazec.

príklad

C++

`"reťazec v \\" jazyku C++"` escape reťazec pre " je \, escape znak je \

Posledným špeciálnym elementom sú zátvorky. Tu je možné definovať zobrazenie pre 3 typy zátvoriek a to obyčajné (,) hranaté [,] a XML tagy <,> (síce tu sa nejedná o zátvorky v pravom slova zmysle, ale spĺňajú rovnakú funkcionalitu). Pri zátvorkách je možné nastaviť ako sa majú zobraziť po nájdení páru. Teda ako sa zvýrazní otváracia a zatváracia zátvorka. Definovanie zobrazenia pozostáva z formátu písma (farba, rez a typ) ale taktiež aj z farby pozadia, pre lepšie vizuálne hľadanie v texte. Ďalej je možné nastaviť, či sa má automaticky doplniť uzatváracia zátvorka v prípade, že sa nenájde jej párová.

3.3 Šablóny

Šablónou budeme rozumieť jednak preddefinovaný text (malá časť zdrojového kódu) určený na vloženie a jednak ďalšie vlastnosti, ktoré s ním súvisia ako počiatočná pozícia kurzora alebo typ šablóny (obyčajná, XML).

Šablóna teda obsahuje :

- Názov šablóny – je názov, ktorý sa zobrazí v kontextovom menu pre použitie šablóny. Zvyčajne ide o skrátenie jej obsahu.
- Farba – ide o farbu akou sa zobrazí názov v menu. Zvyčajne sa volí farba kľúčového slova obsiahnutého v šablóne, ktoré ju najviac vystihuje (napr. pre šablónu for cyklu v C++ by mohlo ísť o farbu kľúčového slova for).
- Počiatočná pozícia kurzora – je pozícia na ktorú sa premiestni kurzor po vložení šablóny. Pozícia sa počíta od začiatku obsahu šablóny.
- Rozlišovanie veľkých a malých písmen – určuje, či pri porovnávaní práve editovaného textu s obsahom šablóny musí nastať presná zhoda s ohľadom na veľkosť písmen.
- Automatické doplnenie – šablóne je možné nastaviť po koľkých zadaných znakov sa má šablóna vložiť automaticky, bez potvrdenia užívateľa.
- Obsah – je samotný text, ktorý sa vloží pri použití šablóny. Tu je nutné rozlíšiť 2 druhy šablón a to či ide o obyčajnú alebo XML šablónu. Pri obyčajnej šablóne sa definuje celý text, ktorý má byť vložený pri použití šablóny. Pri XML šablóne sa definuje iba tag a to či je párový. Samotný vkladaný text sa vygeneruje podľa zvyklostí XML. Pre nepárový tag <tag>, pre párový <tag></tag>.

Použitie šablóny

Šablónu môže užívateľ vložiť z kontextového menu. To sa zobrazí vždy po stlačení klávesy, keď je použitie šablóny možné. Kurzor sa samozrejme musí nachádzať mimo komentárov, reťazcov uzavretých úvodzovkami, apostrofmi a v prípade, že ide o dokument typu XML tak sa kurzor navyše musí nachádzať vnútri niektorého XML tagu. Po vložení šablóny sa kurzor presunie na pozíciu, ktorá je v šablóne definovaná ako počiatočná pozícia kurzoru.

Použitie závisí od druhu šablóny (obyčajná, XML).

Obyčajná šablóna

Pri editovaní sa vezme text oddelený z oboch strán bielymi znakmi, rozoznanými kľúčovými slovami (definovanými v časti so zvýrazňovaním syntaxie) alebo špeciálnymi elementmi (komentáre, čísla, identifikátory, ...) na ľavo od kurzora. Potom nastane zobrazenie kontextového menu obsahujúceho názvy všetkých obyčajných šablón, ktorých obsah začína daným textom. Menu sa samozrejme nezobrazí ak je prázdne. Pri porovnávaní textu so začiatkom obsahu šablóny sa berie ohľad na rozlišovanie veľkých a malých písmen.

XML šablóna

Pri stlačení klávesy "<" dôjde k otvoreniu nového tagu a teda k zobrazeniu kontextového menu so všetkými XML šablónami, pretože je možné doplniť ľubovoľnú z nich. Pri editácii v rámci tagu sa berie text od jeho začiatku, z ktorého sú vynechané počiatočné biele znaky. Ak je takýto text začiatkom niektorého definovaného tagu zobrazí sa jeho šablóna v menu, v ktorom bude na výber. Po stlačení klávesy ">" nastane vyhľadanie začiatku tagu. Ak je nájdený, vezme sa prvé slovo od jeho pozície. V prípade, že je dané slovo zhodné s niektorým definovaným párovým tagom, zobrazí sa jeho šablóna v menu v ktorom bude na výber, pretože je možné doplniť jeho uzatváraciu časť. Pri porovnávaní sa berie ohľad na rozlišovanie veľkých a malých písmen.

Použitie XML šablóny je oveľa flexibilnejšie a nie je možné ho simulovať obyčajnou šablónou. Na záver je nutné povedať, že kontextové menu sa zobrazuje vždy pod pozíciou kurzora okrem prípadov, kedy sa nezmestí na obrazovku, vtedy musí dôjsť ku korekcii pozície. Ak je v danom momente možné použiť aj XML šablónu aj obyčajnú šablónu zobrazia sa samozrejme na výber obe.

3.4 Indentačné pravidlá

Indentácia je proces v ktorom dochádza k vhodnému umiestneniu bielych znakov akými sú medzery a tabulátory na začiatky riadkov za účelom docieliť zvýšenie čitateľnosti zdrojového kódu. Tá sa v tomto editore riadi pravidlami, ktoré definuje užívateľ podľa svojich potrieb. Ide o pravidlá, ktoré sa uplatňujú na základe predchádzajúceho kontextu. Základom pre pravidlo je teda popis situácie, ktorá musí nastať aby došlo k požadovanej indentácii. Je to podobné ako by to musel robiť programátor ak by pracoval v editore bez akejkoľvek indentácie a chcel by zdrojový kód zarovnať podľa svojich predstáv. Jeho práca by vyzerala napríklad nasledovne. Pozrie sa na riadok, ktorý chce odsadiť. Zistí, či sa nachádza v bloku. Ak áno, tak ho odsadí napr. o tabulátor od pozície, kde sa nachádza začiatok bloku a pod. Tento spôsob je pre človeka najprirodzenejší, teda analýza predchádzajúceho kódu a uplatnenie odsadenia. O presne takýto prístup sa snažia aj pravidlá ktorými sa riadi tento editor samozrejme trochu formálnejšie. Nasledujúci text vysvetľuje problematiku tvorby pravidiel a ich vyhodnocovania editorom.

Definovanie pojmov používaných v texte

Kľúčové slovo – slovo, ktoré bolo zadané v časti so zvýrazňovaním syntaxie.
Špeciálny symbol – jeden zo symbolov {…, .B., .N., NOVY_RIADOK}.
Definícia pravidla – zápis pravidla.
Definícia bloku – zápis bloku.
Sila pravidla – počet kľúčových slov obsiahnutých v časti definície pravidla, ktorá obsahuje vyhľadávaný vzor.
<*text*> - označuje miesto, ktoré zadáva užívateľ.
[*text*] – označuje voľbu, ktorá nie je povinná.

Blok

Definícia bloku je nasledovná

BLOK *<začiatok bloku> <koniec bloku>*

<*začiatok bloku*> a <*koniec bloku*> je jedným z kľúčových slov.

Definícia bloku je dôležitá pre špeciálne symboly **...**, **.N.** a **.B.** (viac v časti *Špeciálne symboly*). Zápis bloku musí byť uvedený na jednom riadku, pričom na počet definícií a ich umiestnenie nie je kladené žiadne obmedzenie (začiatok, koniec súboru, nemusia byť uvedené pokope, ani pre použitím špeciálneho symbolu, ktorý ich využíva).

príklad

C/C++

BLOK { }

Pascal

BLOK begin end

BLOK repeat until

Pravidlo

Definícia pravidla je nasledovná

PRAVIDILO_BEGIN [**R**]
<vzor pravidla>
PRAVIDLO_INDENTACIA
<popis indentácie>
PRAVIDLO_END

Pri definovaní pravidiel musí mať užívateľ na pamäti : v prípade, že v danej situácii, je možné uplatniť viacero pravidiel, tak sa uplatní to, ktoré ma väčšiu silu, ak majú pravidla rovnakú silu uplatní sa to, ktoré bolo definované skôr.

[**R**] prítomnosť tejto voľby určuje, či sa bude pravidlo vyhľadávať tzv. do páru. Teda zaistí, že nájdený vzor nie je súčasťou už skôr uplatneného pravidla rovnakej definície, teda inak povedané : nájde sa prvý nespárovaný vzor.

<vzor pravidla> je vzor, ktorý sa bude vyhľadávať pri uplatňovaní pravidla. Pozostáva z kľúčových slov jazyka a špeciálnych symbolov uvedených na samostatných riadkoch. Pred kľúčové slovo alebo špeciálny symbol NOVY_RIADOK je možné uviesť operátor **NOT**, potom sa požaduje na danom mieste jeho absencia.

príklad

Vzor pravidla :

PRAVIDLO_BEGIN
if
(
...
)
PRAVIDLO_INDENTACIA
<indentačná časť >

Je nutné predbehnúť a povedať, že špeciálny symbol ... reprezentuje ľubovoľný text neobsahujúci začiatok ani koniec žiadneho z nadefinovaných blokov. Predpokladajme ešte definíciu jedného bloku :

BLOK { }

Daný vzor odpovedá textu, ktorý má na začiatku kľúčové slovo if, po ktorom nasleduje ľubovoľný počet bielych znakov ukončených slovom (. Ďalej nasleduje ľubovoľný text neobsahujúci začiatok ani koniec bloku, ktorý je ukončený kľúčovým slovom).

Teda daný vzor uspeje napr. na texte

```
1: if (size < 255)
```

Ale neuspeje na textoch

```
1: if (retazec != L""
```

Kôli chýbajúcemu slovu)

```
1: if (retazec { != L"")
```
Kôli výskytu začiatku bloku v texte na ktorý sa snaží nasadiť špeciálny symbol ...

```
1: if else (retazec != L"")
```
Kôli výskytu slova else medzi slovami if a). Medzi nimi sa totiž môžu nachádzať iba biele znaky.

<popis indentácie> určuje indentáciu, ktorá sa uplatní v prípade nájdenia pravidla. Má jeden z nasledujúcich tvarov:

[*<kľúčové slovo>*] **ODKLUCOVEHOSLOVA** *<index kľúčového slova> <indentácia>*

[*<kľúčové slovo>*] **ZACIATOKRIADKA** *<riadok> <offset> <indentácia>*

ODKLUCOVEHOSLOVA – indentovať sa bude od pozície kľúčového slova zadaného parametrom *<index kľúčového slova>*.
ZACIATOKRIADKA - indentovať sa bude od začiatku riadka zadaného parametrami *<riadok>* a *<offset>.*

<indentácia> je reťazec pozostávajúci z písmen M (medzera) a T (tabulátor), v prípade, že sa nemá uplatniť žiadna indentácia, vloží sa písmeno X. Po nájdení pravidla sa indentácia prevedie na reťazec zložený z tabulátorov a medzier. Tá sa následne vloží na miesto určené zvyšnou časťou popisu.
<riadok> nezáporné číslo alebo jeden zo symbolov {^, $}. Symbol ^ označuje prvý riadok textu na ktorom sa našiel vzor uplatneného pravidla, naopak symbol $ označuje riadok posledný. Ak je zadané číslo (n), tak sa berie riadok textu na ktorom sa nachádza n-té kľúčové slovo alebo prvé slovo textu na ktorom uspel n-tý špeciálny symbol vzoru uplatneného pravidla (čísluje sa od 0).
<index kľúčového slova> označuje kľúčové slovo (0 pre prvé slovo) vo vzore pravidla. V prípade, že sa odkazuje na špeciálny symbol, tak sa berie prvé slovo textu na ktorom sa uplatnil.
<offset> je číslo, ktoré sa pripočíta k riadku daného parametrom *<riadok>* ak je určený číslom alebo symbolom ^ (prvý riadok), ak je určený symbolom $ (posledný riadok) tak sa odpočíta.
[*<kľúčové slovo>*] v prípade, že je zadané kľúčové slovo, tak sa pravidlo bude snažiť uplatniť po jeho napísaní na začiatku riadka, inak po vložení nového riadka (stlačení klávesy Enter).

príklad

Indentácia pravidla :

```
<vzor pravidla>
PRAVIDLO_INDENTACIA
                ZACIATOKRIADKA ^ 0 TT
PRAVIDLO_END
```
Pravidlo sa začne hľadať po stlačení klávesy Enter. V prípade nájdenia jeho vzoru sa na daný riadok vloží sekvencia bielych znakov zo začiatku prvého riadka textu na ktorom sa našiel vzor pravidla ku ktorým sú navyše pridané dva tabulátory.

```
<vzor pravidla>
PRAVIDLO_INDENTACIA
                else ODKLUCOVEHOSLOVA 0 X
```

PRAVIDLO_END

Pravidlo sa začne hľadať po napísaní kľúčového slova else na začiatku riadka. V prípade nájdenia jeho vzoru sa na začiatok riadka vloží toľko tabulátorov a medzier aká je vzdialenosť prvého kľúčového slova alebo špeciálneho symbolu vzoru pravidla od začiatku riadka na ktorom sa nachádza v texte na ktorom sa našiel vzor. Vzdialenosťou sa myslí dĺžka textu od začiatku riadka po dané kľúčové slovo alebo špeciálny symbol. Čo sa týka počtu vložených medzier a tabulátorov, tak tabulátorov sa vloží toľko koľko je ich obsiahnutých v texte od začiatku riadka po dané kľúčové slovo alebo špeciálny symbol. Zvyšok je nahradený medzerami. Tým sa dosiahne presné zarovnanie.

Špeciálne symboly

V tejto časti sa pojmom blok rozumie blok, ktorý užívateľ nadefinoval podľa popisu vyššie.

Význam špeciálnych symbolov
NOVY_RIADOK reprezentuje koniec jedného riadka a začiatok druhého.
... reprezentuje ľubovoľný text, ktorý neobsahuje začiatok ani koniec žiadneho bloku.
.N. reprezentuje ľubovoľný text, ktorý neobsahuje začiatok ani koniec žiadneho bloku a nachádza sa na jednom riadku.
.B. reprezentuje ľubovoľný text, ktorý obsahuje správne uzavreté bloky.

Ku všetkým symbolom okrem **NOVY_RIADOK** je možné pridať operátor **NOT**, ktorý má nasledovný zápis

NOT *<zoznam kľúčových slov>*

potom je význam špeciálneho symbolu rozšírený o to, že text, ktorý daný symbol reprezentuje nesmie obsahovať žiadne z kľúčových slov obsiahnutých v zozname.
<zoznam kľúčových slov> je zoznam kľúčových slov oddelených medzerami alebo tabulátormi.

K symbolom **...** a **.N.** je možné pridať ľubovoľný počet operátorov **PAR**, ktoré majú nasledujúci zápis

PAR *<začiatok páru> <koniec páru>*

potom je význam špeciálneho symbolu rozšírený o to, že text, ktorý daný symbol reprezentuje musí obsahovať správne spárované páry.
<začiatok páru> a *<koniec páru>* je jedným z kľúčových slov.
Operátory **PAR** musia byť uvedené pred **NOT**, pretože inak by sa **PAR** bralo ako kľúčové slovo zo zoznamu slov pre **NOT**.

príklad

... PAR ( ) **PAR** [ ] **NOT** ;

reprezentuje text, ktorý neobsahuje bloky, má správne spárované zátvorky a neobsahuje kľúčové slovo ; (čo ide napr. v C++ interpretovať ako text ktorý je súčasťou jedného príkazu)

Nakoniec uvediem zopár príkladov indentácií ktoré sú známe z praxe.

Pre klasickú blokovú indentáciu C++ stačí uviesť nasledujúce definície.

BLOK { }

```
PRAVIDLO_BEGIN
        {
        .B.
PRAVIDLO_INDENTACIA
        ZACIATOKRIADKA 0 0 T
PRAVIDLO_END

PRAVIDLO_BEGIN
        {
        .B.
PRAVIDLO_INDENTACIA
        } ZACIATOKRIADKA 0 0 X
PRAVIDLO_END
```

Na začiatok uvedieme definíciu bloku, ktorého začiatok tvorí kľúčové slovo { a koniec }. Pre blokovú indentáciu je charakteristické to, že vo vnútri bloku sa odsadzuje o napr. tabulátor od začiatku riadka na ktorom sa nachádza jeho začiatok. Presne to vystihuje definícia prvého pravidla. Prvé pravidlo hovorí, že po nájdení začiatku bloku od ktorého sú všetky bloky správne spárovane (vyhľadávaný vzor) sa má indentovať o tabulátor (T) od začiatku riadka na ktorom sa nachádza prvé slovo vzoru pravidla (ZACIATOKRIADKA 0 0). Prvé pravidlo nám teda zaručuje indentáciu v blokoch, ale pre úplnosť je potrebné ešte pridať pravidlo, ktoré po napísaní konca bloku na začiatok riadka zarovná riadok na úroveň riadka s jeho začiatkom. O to sa stará druhá definícia.

Nasledujúce príklady zabezpečujú niektoré funkcie z tzv. smart indentácie pre jazyk C++. Celkový popis by bol zdĺhavý a obsahoval by desiatky pravidiel, preto uvediem len pár zaujímavých situácií a to indentáciu if-else, indentáciu zátvoriek a príkazov.

Pre nasledujúce pravidlá sa predpokladá definícia bloku

```
BLOK { }
```

Indentácia if-else

```
PRAVIDLO_BEGIN R
        if
        .B.
PRAVIDLO_INDENTACIA
        else ODKLUCOVEHOSLOVA 0 X
PRAVIDLO_END
```

Uvedené pravidlo zaistí, že po napísaní kľúčového slova else na začiatku riadka sa začne hľadať najbližšie kľúčové slovo if od ktorého nasledujú správne uzavreté bloky. Po nájdení vzoru sa else zarovná na pozíciu if. Pravidlo má nastavený prepínač R, ktorý zaistí, že nájdené if nie je súčasťou vzoru pre už skôr uplatnené pravidlo rovnakej definície, teda inak povedané : nájde sa prvé nespárované if s else.

Indentácia zátvoriek

```
PRAVIDLO_BEGIN
        (
        ... PAR ( ) PAR [ ] NOT ;
PRAVIDLO_INDENTACIA
        ) ODKLUCOVEHOSLOVA 0 X
PRAVIDLO_END

PRAVIDLO_BEGIN
        (
        ... PAR ( ) PAR [ ] NOT ;
PRAVIDLO_INDENTACIA
        ODKLUCOVEHOSLOVA 0 T
PRAVIDLO_END
```

Uvedené pravidlá umožňujú indentáciu obyčajných zátvoriek. Pre hranaté zátvorky by pravidlá vyzerali rovnako až na to že by sa museli v ich definíciách zameniť obyčajné zátvorky za hranaté a naopak. Prvé pravidlo zaistí umiestnenie) na úroveň prvej nespárovanej (. Vo vzore pravidla sa požaduje aby text od nájdenej otváracej zátvorky mal správne spárované zátvorky oboch typov a neobsahoval kľúčové slovo ;. Požaduje sa to preto, aby sa našla otváracia zátvorka, ktorá skutočne tvorí pár so zátvorkou po ktorej napísaní sa snaží dané pravidlo uplatniť a zároveň aby nájdený text bol súčasťou jedného príkazu. Ak by sme vynechali NOT ;, tak by mohlo dôjsť k párovaniu zátvoriek obsiahnutých vo viacerých príkazoch. Druhé pravidlo hovorí, že vo vnútri obyčajných zátvoriek sa po stlačení klávesy enter riadok odsadí o tabulátor od pozície otváracej.

Indentácia príkazu

```
PRAVIDLO_BEGIN
      ;
      NOVY_RIADOK
      ...
      ;
PRAVIDLO_INDENTACIA
      ODKLUCOVEHOSLOVA 2 X
PRAVIDLO_END

PRAVIDLO_BEGIN
      ;
      NOVY_RIADOK
      .N.
      NOVY_RIADOK
 PRAVIDLO_INDENTACIA
      ZACIATOKRIADKA $ 0 T
PRAVIDLO_END

PRAVIDLO_BEGIN
      NOVY_RIADOK
      .N.
      NOVY_RIADOK
PRAVIDLO_INDENTACIA
      ZACIATOKRIADKA $ 0 X
PRAVIDLO_END
```

Nasledujúce pravidlá zaisťujú, že každý nový príkaz bude odsadený rovnako ako predchádzajúci. Pokračovanie príkazu na druhom riadku bude odsadené o tabulátor od začiatku príkazu a pokračovanie na ďalších riadkoch bude kopírovať odsadenie predchádzajúceho riadka. Prvé pravidlo spôsobí zarovnanie nového príkazu na úroveň predchádzajúceho a to tak, že sa nájde text obklopený bodkočiarkami, teda text, ktorý tvorí predchádzajúci príkaz. Následne sa zarovnáva podľa prvého slova príkazu. Druhé pravidlo zaistí odsadenie pokračovania príkazu na druhom riadku o tabulátor. A nakoniec posledné pravidlo zaistí, že každé ďalšie pokračovanie príkazu bude odsadené ako predchádzajúci riadok. Za povšimnutie stojí, že posledné pravidlo síce uspeje na každom texte majúcom 2 riadky a neobsahujúcom bloky, ale kvôli nulovej sile sa uplatní len zriedkavo. Tieto pravidlá nie sú kompletné. Neriešia napr. situáciu kedy chceme indentovať po prvom príkaze od začiatku bloku a pod. Ale slúžia ako ukážka indentácie v postupnosti príkazov.

3.5 Inštalácia programu

Inštalácia prebieha nakopírovaním programu so všetkými jeho súčasťami do ľubovoľného adresára. Dôležité je, aby všetky súbory, ktoré program využíva k svojej

činnosti boli umiestnené v rovnakom adresári ako program samotný, pretože práve takéto rozmiestnenie editor predpokladá.

3.6 Konfiguračné súbory

Všetky konfiguračné súbory sú uložené v textových súboroch (.txt) takže je možná aj ich ručná editácia, resp. úprava v inom editore, čo môže byť vhodné napr. pre súbory obsahujúce indentačné pravidlá, ale nie je to doporučený spôsob, keďže zmeny v jednom súbore si niekedy vyžadujú aktualizáciu na viacerých miestach napr. premenovanie štýlu, ... Preto je vhodnejšie akékoľvek zmeny realizovať pomocou dialógov na to určených.

Zoznam súborov

AEdit.styly – obsahuje zoznam vytvorených štýlov.

Pre každý štýl sú vytvorené nasledujúce súbory (Štýl zastupuje názov niektorého existujúceho štýlu napr. C++) :

- Štýl.pravidla – obsahuje užívateľom definované indentačné pravidlá.
- Štýl.sablony – obsahuje užívateľom definované šablóny.
- Štýl.skupiny – obsahuje formátovanie skupín, ktoré obsahujú kľúčové slová.
- Štýl.specialneformatovanie – obsahuje formátovanie špeciálnych elementov ako komentáre, identifikátory, čísla a pod.
- Štýl.zatvorkyxml – obsahuje formátovanie zátvoriek po nájdení páru, nastavenie doplňovania v prípade jeho nenájdenia a to, či bude daný štýl uplatňovaný na dokumentoch so štruktúrou XML.
- Štýl.vlastnosti – obsahuje základné vlastnosti štýlu ako popis, zoznam prípon pre súbory obsahujúce zdrojový kód, aplikácia, ktorej môže byť editovaný súbor predaný na ďalšie spracovanie a počiatočný text, ktorý sa vloží pri vytvorení nového dokumentu.
- tmp.prvapripona – prvá prípona je prvou príponou zo zoznamu zadaného vo vlastnostiach štýlu. Tento súbor je vytvorený ak dôjde k predaniu neuloženého súboru na spracovanie inej aplikácii (zadanej vo vlastnostiach štýlu).

3.7 Popis užívateľského prostredia

Popis prostredia (menu, toolbar a stavový riadok).

Menu

Menu obsahuje :

- podmenu *File*
  New – zobrazenie dialógu na zvolenie štýlu v ktorom bude vytvorený nový dokument.
  Open … - otvorenie existujúceho dokumentu.
  Close – zatvorenie dokumentu.
  Save – uloženie dokumentu.
  Save As – uloženie dokumentu pod iným menom.
  Print ... – tlačenie dokumentu.

Print Preview – zobrazenie náhľadu dokumentu pred tlačením.
Print Setup – zobrazenie dialógu s nastaveniami pre tlačiareň.
Zobrazenie posledných 4 otvorených dokumentov - kliknutím na niektorý z nich dôjde k jeho otvoreniu.
Exit – ukončenie programu.

- podmenu *Edit*
  Undo – vrátenie dokumentu do stavu pred poslednou úpravou textu.
  Cut – vystrihnutie textu.
  Copy – kopírovanie textu.
  Paste – vloženie textu.
  Select All – označenie celého textu.
  Find – zobrazenie dialógu na vyhľadávanie textu.
  Find Next – nájdenie ďalšieho výskytu textu zadaného na hľadanie.
  Replace – zobrazenie dialógu pre nahradzovanie textu.
  Indentovať – uplatnenie indentačných pravidiel na všetky riadky obsiahnuté v označenej časti textu.

- podmenu *View*
  Toolbar – nastavenie viditeľnosti toolbaru.
  StatusBar – nastavenie viditeľnosti statusbaru.

- podmenu *Projekt*
  Zmeniť štýl – zobrazenie dialógu na výber zo všetkých existujúcich štýlov. Po zvolení niektorého z nich dôjde k preformátovaniu dokumentu (podľa zvoleného štýlu).
  Vlastnosti – zobrazenie dialógu s vlastnosťami práve editovaného dokumentu.
  Predať inej aplikácii – spustenie aplikácie priradenej k štýlu, v ktorom je zobrazený editovaný dokument a predanie jeho obsahu na spracovanie.

- podmenu *Štýly*
  Editovať – zobrazenie dialógu na editovanie štýlov.

- podmenu Window
  Cascade, Tile, Arrange Icons – usporiadanie okien podľa zvoleného typu.
  Zoznam všetkých otvorených dokumentov – kliknutím na niektorý z nich dôjde k jeho zaktívneniu.

- podmenu Help
  About AEdit... – zobrazenie dialógu obsahujúceho základné informácie o programe.

<u>Toolbar</u>

Z toolbaru sú dostupné nasledujúce funkcie :
- Vytvorenie nového dokumentu
- Otvorenie existujúceho dokumentu
- Uloženie editovaného dokumentu
- Vystrihnutie textu
- Kopírovanie textu

- Vloženie textu
- Tlačenie dokumentu
- Informácie o programe

Stavový riadok

Okrem štandardného zobrazenia správ pri označení položiek v menu a toolbare v časti 1, zapnutia kláves ako NUMLOCK, CAPSLOCK v časti 6. Boli do stavového riadka pridané informácie o poslednom použitom indentačnom pravidle (2), o použitom štýle (3), o pozícii kurzora (4) a veľkosti aktuálneho označenia (5) (počet znakov označeného textu). Tieto informácie sú zobrazené pre práve editovaný dokument, teda ten, ktorého okno je aktívne.

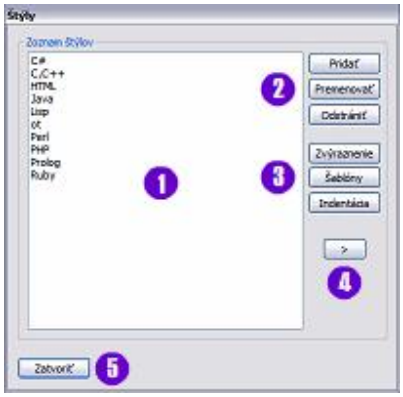
Vytvorenie nového dokumentu

Je možné z hlavného menu File/New, toolbaru, alebo pomocou klávesovej skratky Ctrl + N. Pred jeho samotným vytvorením sa najprv zobrazí dialóg na zvolenie štýlu, ktorý sa na ňom uplatní. V prípade, že užívateľ chce editovať iba obyčajný text musí zvoliť štýl ot (pre obyčajný text). Ide o preddefinovaný prázdny štýl, ktorý vynikol práve pre tento účel. Po označený niektorého zo štýlov v zozname (1) sa zobrazí jeho popis pre lepšiu orientáciu (2). Napr. ak máme nadefinovaných viac štýlov pre jeden programovací jazyk je popis vhodný na ich odlíšenie. Tlačítkom Použiť (3) dôjde k vytvoreniu nového dokumentu s použitím označeného štýlu. Tlačítkom Zrušiť (4) nastane zrušeniu celého procesu, teda k vytvoreniu nového dokumentu nedôjde.

Otvorenie existujúceho dokumentu

Je možné z hlavného menu File/Open, toolbaru, alebo pomocou klávesovej skratky Ctrl + O. Pri otváraní sa najprv overí, či prípona súboru nie je priradená k niektorému štýlu. Ak áno tak sa naň automaticky uplatní. V prípade ak prípona nie je rozpoznaná je na otváraný súbor uplatnený štýl pre obyčajný text.

Editovanie štýlov



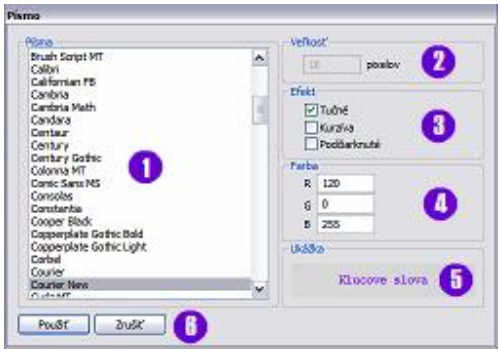
Sa realizuje pomocou dialógu na obrázku. Ten je dostupný z hlavného menu Štýly/Editovať. Pomocou tlačítok Pridať, Premenovať a Odstrániť (2) je možné pridať nový štýl, premenovať alebo odstrániť existujúci (najprv je nutné označiť niektorý zo zoznamu (1)). Pomocou tlačítok Zvýraznenie, Šablóny a Indentácia (3) sa pristupuje k dialógom na editovanie zvýrazňovania, šablón a indentačných pravidiel označeného štýlu. Tlačítko > (4) sprístupňuje časť so základnými vlastnosťami štýlu ako jeho popis, zoznam

prípon súborov nesúcich zdrojový kód pre daný štýl, aplikácia, ktorej sa možné predať editovaný dokument na ďalšie spracovanie, farba pozadia pri editovaní a počiatočný text, ktorý bude vložený pri vytvorení nového dokumentu. Zatvorenie dialógu nastane po stlačení tlačítka Zatvoriť (5).

Editovanie zvýrazňovania

Sa realizuje pomocou dialógu na obrázku. Ten je dostupný z dialógu na editovanie štýlov. V časti 1 je možné pridávať, premenovávať a odstraňovať skupiny. Po označený niektorej z nich sa v časti 2 zobrazia slová v nej obsiahnuté. Tie je možné pridávať a odstraňovať. V časti 3 je zobrazený náhľad formátovania slov označenej skupiny. Formátovanie je možné upraviť pomocou tlačítka Zmeniť, ktoré vyvolá dialóg na nastavenie písma. V časti 4 je zobrazená funkcia danej skupiny a teda či ide o skupinu oddeľovačov. V časti 5 je možné nastaviť, či sa v danom štýle majú rozlišovať veľké a malé písmená v kľúčových slovách. Ďalej sa dá v dialógu určiť či sa bude editovaný štýl uplatňovať na dokumentoch so štruktúrou XML (v časti 7). Nakoniec je možné pristúpiť k dialógom na editovanie špeciálnych elementov a zátvoriek daného štýlu v časti 6. Nastavenia v častiach 4, 5 a 7 je možné meniť zaškrtnutím resp. odškrtnutím príslušnej voľby. Všetky uskutočnené zmeny je nutné potvrdiť tlačítkom Uložiť. Tlačítkom Zrušiť dôjde k zatvoreniu dialógu a strate všetkých uskutočnených úprav.

Nastavenie písma



Sa realizuje pomocou dialógu na obrázku. V dialógu je možné nastaviť : typ písma jeho označením v zozname (1), veľkosť písma v pixeloch (2) (ale len pre element obyčajný text), efekt písma, teda, či sa má písmo zobrazovať tučne, kurzívou alebo má byť podčiarknuté zaškrtnutím danej voľby (3) a farbu písma v modely RGB (4). Na priebežnú kontrolu výsledného formátovania slúži ukážka (5), ktorá sa aktualizuje po každej zmene. Pre uplatnenie nastavení je nutné ukončiť dialóg pomocou tlačítka Použiť (v časti 6). Naopak ak nastavenie nemá byť uplatnené je nutné dialóg ukončiť tlačítkom Zrušiť (v časti 6).

Nastavenie farby

Sa realizuje pomocou dialógu na obrázku. Dialóg slúži na nastavenie farby písma, alebo farby jeho pozadia (podľa situácie). Farba sa zadáva v modely RGB (2). Akákoľvek zmena v nastavení farby sa prejaví v náhľade (1) obsahujúcom text, pre ktorý sa volí formátovanie. Pre

uplatnenie nastavení je nutné ukončiť dialóg pomocou tlačítka Použiť (3). Naopak ak nastavenie nemá byť uplatnené je nutné dialóg ukončiť tlačítkom Zrušiť (4).

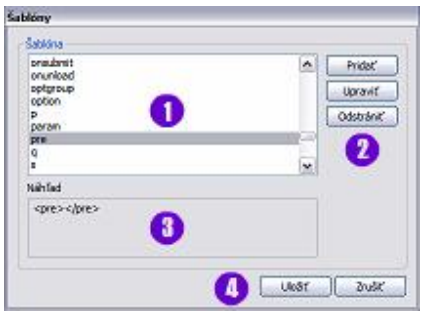
Zátvorky a XML elementy

Ich nastavnie sa realizuje pomocou dialógu na obrázku dostupného z dialógu pre editovanie zvýrazňovania štýlu. Dialóg slúži na nastavenie zobrazenia po nájdení páru a nastavenie automatického doplňovania v prípade jeho nenájdenia pre každý z troch typov zátvoriek. Zobrazovanie po nájdení páru a automatické doplňovanie sa nastaví zaškrtnutím odpovedajúcej voľby u daného typu zátvoriek (3). Nastavenie formátovania nájdeného páru sa realizuje pomocou dialógov pre nastavenie písma a farby , ktoré sú prístupné tlačítkami Písmo a Pozadie (2). Pri zmene formátovania nastane automaticky aktualizácia náhľadu (1). Pre uplatnenie nastavení je nutné ukončiť dialóg pomocou tlačítka Uložiť (v časti 4). Naopak ak nastavenie nemá byť uplatnené je nutné dialóg ukončiť tlačítkom Zrušiť (v časti 4).

Editovanie špeciálnych elementov

Na tento účel slúži dialóg na obrázku dostupný z dialógu pre editovanie zvýrazňovania štýlu. V dialógu sa nastavuje zobrazovanie špeciálnych elementov ako obyčajný text (1), identifikátory, čísla, reťazce uzavreté úvodzovkami a apostrofmi, jednoriadkový a viacriadkový komentár. U každého z elementov sa zadáva formátovanie písma pomocou tlačítka písmo. Pri jednoriadkovom komentári (5) je nutné zadať reťazec, ktorý ho uvádza. V prípade, že ostane textové políčko prázdne tak to znamená, že sa nebude pri editovaní dokumentu vyhľadávať. Pri viacriadkovom komentári (6) je nutné zadať jeho začiatok a koniec. Ak nie sú zadané obe políčka, tak sa viacriadkový komentár pri editovaní nebude vyhľadávať. Pri reťazcoch uzavretých úvodzovkami alebo apostrofmi (4) je možné zakázať ich formátovanie, čoho dôsledkom bude to, že ich obsah bude zobrazovaný. Taktiež je možné zadať reťazce stojace vpravo a vľavo, ktoré rušia význam apostrofu a úvodzovky. Pri číslach (3) a identifikátoroch (2) je možné nastaviť ich zobrazovanie. Ak je zobrazovanie vypnuté, tak sa formátujú ako obyčajný text. U identifikátora je možné zadať zoznam znakov, ktoré sa v ňom môžu vyskytnúť navyše okrem písmen a číslic. Nakoniec ostáva zadanie escape znaku (7) (je možné zadať reťazec, ale obvykle sa jedná iba o jeden znak). Ten je zvyčajne totožný z escape sekvenciou pre apostrof a úvodzovku. Jeho zadanie je nutné pre správne zobrazovanie reťazcov (samozrejme ak ho daný programovací jazyk obsahuje).

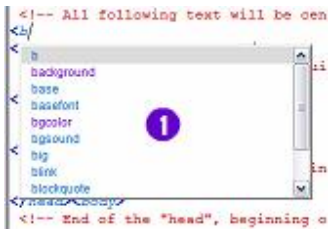
Šablóny



Na manipuláciu s nimi slúži dialóg na obrázku, ktorý je dostupný z dialógu pre editovanie štýlov. Pomocou tlačítka Pridať (v časti 2) sa zobrazí dialóg na pridanie novej šablóny. Pomocou tlačítka Upraviť (v časti 2) sa zobrazí dialóg na editovanie označenej šablóny. Tlačítkom Odstrániť (v časti 2) dôjde k odstráneniu označenej šablóny. Po označení niektorej zo šablón (1) sa zobrazí v časti 3 náhľad jej obsahu. V prípade, že ide o XML šablónu, tak je obsah najprv vygenerovaný z jej vlastností. Všetky uskutočnené zmeny je nutné potvrdiť tlačítkom Uložiť. Tlačítkom Zrušiť dôjde k zatvoreniu dialógu a strate všetkých uskutočnených úprav.

Pridanie a upravenie šablóny

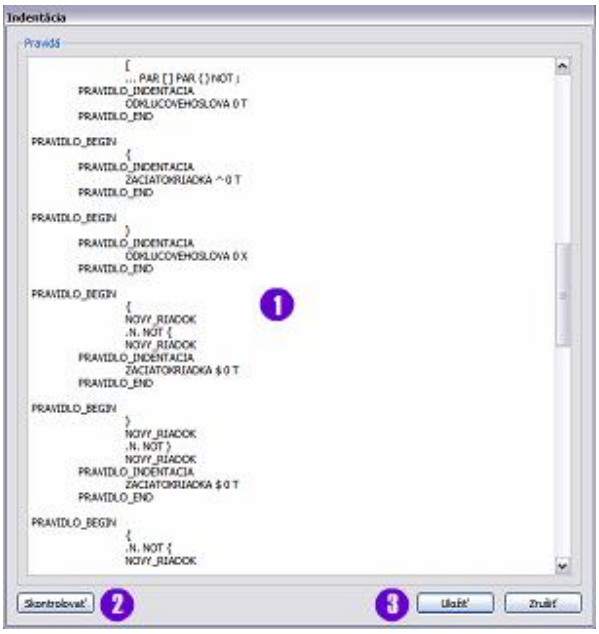
Šablóny je možné pridávať a upravovať pomocou dialógu na obrázku, ktorý je dostupný z dialógu pre editovanie šablón. Tento dialóg sa pre obe operácie líši názvom v titule okna a názvom tlačítka pre úspešné ukončenie operácie (pridanie : "Pridať šablónu", "Pridať" upravenie : "Upraviť šablónu", "Upraviť"). Pre šablónu je možné nastaviť je názov (2). Ten sa zobrazí v kontextovom menu pre doplňovanie šablón. Farbu názvu šablóny pomocou tlačítka Farba (1). Ide o farbu, ktorou sa zobrazí názov v menu. Pozíciu kurzora po doplnení šablóny (2) (zvyčajne ide o miesto kde sa očakáva vstup užívateľa). Zaškrtnutie voľby o nerozlišovaní veľkých a malých písmen (2) spôsobí, že pri analýze uplatnenia šablóny nebude braný ohľad na veľké a malé písmená. Automatické doplnenie po n (zadané užívateľom do príslušného textového poľa v časti 2) znakoch znamená, že šablóna bude vložená po napísaní n znakov z jej obsahu. Obsah šablóny je rozlíšený pre 2 typy a to pre obyčajnú šablónu a XML šablónu. Pre obyčajnú šablónu sa zadáva text (3), ktorý pri použití bude vložený presne tak ako sa zadal. Pre XML šablónu sa zadáva iba tag a to či je párový (v časti 4). Na základe toho sa vygeneruje obsah, ktorý bude vložený po uplatnení šablóny. Pri zmena názvu šablóny alebo jeho farby je automaticky aktualizovaný náhľad v časti 1, ktorý znázorňuje ako bude vyzerať šablóna po jej zobrazení v menu. Pre uplatnenie nastavení je nutné ukončiť dialóg pomocou tlačítka Použiť (alebo Upraviť, podľa situácie) (v časti 5). Naopak ak nastavenie nemá byť uplatnené je nutné dialóg ukončiť tlačítkom Zrušíť (v časti 5).



Nasledujúci obrázok ukazuje zobrazenie kontextového menu (1), pre dopĺňanie šablón. V menu je možný pohyb pomocou kláves : šípka hore, šípka dole, home, end, page up, page down. Vloženie označenej šablóny nastane po stlačení klávesy enter. Zatvorenie menu nastáva po stlačení klávesy escape alebo šípiek doľava a doprava.

Indentačné pravidlá

Indentačné pravidlá sa editujú v dialógu na obrázku, ktorý je dostupný z dialógu pre editovanie štýlov. Časť 1 slúži na ich zadávanie. Pomocou tlačítka Skontrolovať (2) je možné overiť správnosť ich syntaxie. V prípade, že pravidlá nie sú v zadané korektne zobrazí sa správa obsahujúca prvú nájdenú chybu a riadok na ktorom sa vyskytla. Ak sú pravidlá zadané správne, zobrazí sa takisto správna oznamujúca ich korektnosť. Všetky uskutočnené zmeny je nutné potvrdiť tlačítkom Uložiť (v časti 3). Tlačítkom Zrušiť (v časti 3) dôjde k zatvoreniu dialógu a strate všetkých uskutočnených úprav.



4 Vývojová dokumentácia

4.1 Editor

Editor bol naprogramovaný v jazyku C++ prostredníctvom MFC (Microsoft Foundation Classes). Ide o MDI aplikáciu (umožňuje prácu s viacerými dokumentmi súčasne) , ktorá pre zobrazovanie textu s rôznymi vlastnosťami ako farba, veľkosť písma, ... používa komponentu CRichEditView a jej formát RTF v ktorom ukladá text. Tá je predkom pre triedu, ktorá zabezpečuje funkcie ako zvýrazňovanie syntaxie, indentácia, doplňanie zátvoriek, kľúčových slov a XML elementov.

4.2 Zvýrazňovanie syntaxie

Načítanie štýlu

Na začiatku editovania dokumentu (pri jeho otvorení alebo vytvorení nového) dochádza k načítaniu zvýrazňovania. Postupne sa načítavajú všetky skupiny. Pre každé slovo v skupine sa vygeneruje unikátny token, ktorý je predaný konečnému automatu. Ten zaistí svoje rozšírenie o dané slovo tak aby pri jeho rozoznaní vrátil požadovaný token.

Do automatu sa takisto pridajú : reťazec uvádzajúci jednoriadkový komentár, reťazce ohraničujúce viacriadkový komentár, úvodzovka, apostrof, zoznam znakov pre identifikátor.
Automat zaistí vytvorenie častí rozoznávajúcich :

- komentáre – je nutné poznamenať, že automat nerozoznáva viacriadkový komentár tak ako sa očakáva a to ako celok. Automat najprv nájde reťazec ktorým sa komentár začína, potom analyzuje jeho obsah (ten sa nezobrazí pretože je v jeho vnútri) až kým nenájde jeho koniec. Je to kvôli tomu, aby nebolo nutné po zrušení komentára aktualizovať informácie o riadkoch, ktoré sa v ňom nachádzali (bližšie popísané v časti o spracovaní viacriadkového komentára).

  príklad

```
1: /*
2: if (spracuj() == -1){
3:   return;
4: }
5: */
```

  V tomto prípade sa nájde (za predpokladu, že je nadefinované kompletné zobrazenie pre jazyk C++) :
  Na riadku 1 : začiatok komentára /*
  Na riadku 2 : if, (, identifikátor, (,), ==, -, 1, {
  Na riadku 3 : return, ;
  Na riadku 4 : koniec kometára */
  Ak by nebolo nutné analyzovať obsah našiel by sa jedine komentár.

- reťazce uzavreté úvodzovkami alebo apostrofmi.
- identifikátory tvaru [a-zA-Z”zoznam znakov”][0-9a-zA-Z”zoznam znakov”]* kde zoznam znakov obsahuje znaky zadané užívateľom.

príklad

identifikátor v C++ a Pascale
[a-zA-Z_][0-9a-zA-Z_]*, zoznam znakov je [_].

```
1: string _nazovTokenu = "komentar";
2: if (_nazovTokenu.GetLength() >= MAX_SIZE){
```

Ako identifikátory sa rozoznajú : string, _nazovTokenu, GetLength a MAX_SIZE.

- čísla
  [0-9]+ (celé čísla v desiatkovej sústave napr. 23),
  0[xX][0-9a-fA-F]+ (celé čísla v šestnástkovej sústave napr. 0x12b),
  [0-9]+\.[0-9]+ (reálne čisla s pevnou rádovou čiarkou napr. 3.14),
  [0-9]+\.[0-9]+[eE][-+]\{0,1\}[0-9]+ (reálne čísla v semilogaritmickom tvare napr. 56.12E-2).

Pri načítavaní skupín, komentárov, identifikátorov, ... dochádza takisto ku generovaniu RTF kódu, ktorý odzrkadľuje ich formátovanie. Tento kód sa pri zvýrazňovaní textu vkladá do RTF streamu, ktorý je následne vložení do komponenty (editora).

Analýza pomocou konečného automatu

Po načítaní zvýrazňovania sa vytvorí konečný automat, ktorý je základom pre zvýrazňovanie syntaxie. Jeho úlohou je previesť na vstupnom texte lexikálnu analýzu, teda rozložiť vstup na postupnosť tokenov. Z tejto postupnosti sa následne určí, či dané kľúčové slovo patriace tokenu je možné zobraziť. Pri analýze sa ignorujú nadbytočné medzery a tabulátory. Automat vždy vracia token najdlhšej zhody.

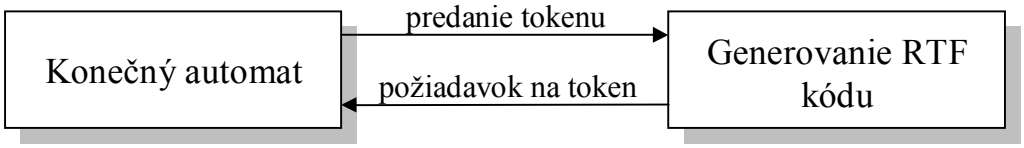
príklad

ak sú nadefinované slová if, ifdef a # tak pri analýze

```
#ifdef MACRO
```

sa vrátia iba tokeny # a ifdef hoci po # sa našlo prvé if.
(v skutočnosti by sa nadefinovali iba 2 kľúčové slová a to if a #ifdef, ale kôli príkladu sa to urobilo takto)

Priebeh zvýrazňovania znázorňuje nasledujúci obrázok.



Po každom nájdení nejakého lexému automatom v analyzovanom texte je vrátený token, ktorý je následne spracovaný časťou programu starajúcej sa o zvýrazňovanie. Najprv sa zistí či daný token reprezentuje kľúčové slovo. Ak áno zistí sa, či je súčasťou skupiny oddeľovačov, pretože takéto slovo je zvýraznené v ľubovoľnom kontexte (mimo komentárov a reťazcov). Ak kľúčové slovo nie je oddeľovačom zvýrazní sa iba v prípade ak stojí v okolí 1. bielych znakov, 2. komentárov a reťazcov (ale nie je ich súčasťou) alebo 3. kľúčových slov, ktoré sú oddeľovačmi. Pri jeho nájdení sa najprv overí ľavá strana, pretože tá už bola zanalyzovaná. Ak je v poriadku tak sa slovo pozdrží do predania ďalšieho tokenu, podľa ktorého sa rozhodne o jeho zobrazení (overenie pravej strany). Zobrazovanie prebieha tak že sa do postupne generovaného RTF streamu vloží RTF kód obsahujúci formátovanie skupiny, do ktorej dané slovo patrí na pozíciu pred začiatok slova.

V prípade, že sa nejedná o kľúčové slovo musí nastať niektorá zo situácií.
Nájde sa

- jednoriadkový komentár – zvýrazní sa bez ohľadu na kontext
- viacriadkový komentár - zvýrazní sa bez ohľadu na kontext
- úvodzovka, apostrof - zvýrazní sa bez ohľadu na kontext
- číslo, identifikátor – pri zvýrazňovaní sa uplatnia pravidlá ako pri zvýrazňovaní kľúčového slova, ktoré nie je oddeľovačom.
- neznámy lexém, teda postupnosť znakov v analyzovanom texte, ktorá neprodukuje žiady token. Zvýrazní sa bez ohľadu na kontext, pričom na jeho zobrazenie je použité formátovanie písma pre obyčajný text.

Nakoniec je vygenerovaný kód vložený (nastreamovaný) do komponenty na miesto kam patrí.

Zvýrazňovanie

Zvýrazňovanie prebieha vždy na texte, ktorý pozostáva z celých riadkov. Kvôli rýchlosti dochádza k zvýrazňovaniu iba viditeľnej časti obrazovky. Aj to iba oblasti od jej prvého po posledný nesformátovaný riadok. Bolo by možné formátovať len jednotlivé riadky resp. súvislé oblasti, ktoré to potrebujú, ale keďže farbenie je realizované pomocou vkladania RTF kódu, je rýchlejšie ak sa to udeje len raz, hoci sa budú zbytočne formátovať riadky, ktoré to nepotrebujú. Aby bolo možné určiť, ktorý riadok už sformátovaný je a ktorý na to ešte len čaká musí sa každému z nich nastaviť príznak aktuálnosti/neaktuálnosti zvýraznenia. Teda k formátovaniu dochádza iba v prípade, že sa na obrazovke nachádza aspoň jeden nesformátovaný riadok. To prináša výhody hlavne pre viacriadkové komentáre, kedy napr. po zrušení komentára na začiatku dokumentu by bolo nutné jeho preformátovanie po jeho koniec, takto postačuje iba preformátovať viditeľnú časť a zvyšným riadkom nastaviť príznak o neaktuálnosti ich zvýraznenia a ich formátovanie nechať na dobu kedy to bude nutné. Teda kedy dôjde k ich zobrazeniu (napr. scrollovaním).

K zvýrazňovaniu dochádza :

- Pri otvorení alebo vytvorení nového dokumentu – dochádza k zvýrazneniu viditeľnej časti obrazovky, zvyšným riadkom do konca dokumentu sa nastaví príznak o ich nesformátovaní.

- Pri scrollovaní – je preformátovaný text nachádzajúci sa medzi prvým a posledným nesformátovaným riadkom viditeľnej časti. K scrollovaniu môže dôjsť prostredníctvom kláves, scrollbaru a myši.

- Pri zmenách na jednom riadku – dochádza k preformátovaniu celého riadka. Ak došlo na riadku k napísaniu alebo zmazaniu viacriadkového komentára, resp. k jeho odkrytiu alebo zakrytiu

  príklad

  zmazanie viacriadkového komentára

```
pred : for (;;){ /* komentar
po   : for (;;){ / komentar
```

  pridanie viacriadkového komentára

```
pred : if (i == 23) break;
po   : /*if (i == 23) break;
```

odkrytie viacriadkového komentára

```
pred : string s = "/* komentar";
po   : string s = /* komentar";
```

zakrytie viacriadkového komentára

```
pred : while (nChar == 'a'){ /* komentar
po   : while (true) //(nChar == 'a'){ /* komentar
```

tak sa musí určiť po ktorý riadok je nutné preformátovať text (bližšie vysvetlenie v *Spracovaní viacriadkových komentárov*). Nakoniec v prípade ak ide o XML dokument je nutné zistiť, či nedošlo k otvoreniu alebo zatvoreniu nejakého tagu, ktorý sa rozkladá na viacerých riadkoch. V takom prípade je nutné preformátovanie po najbližší nižší riadok obsahujúci začiatok, alebo koniec nejakého tagu. Dostávame tri oblasti, ktoré je nutné preformátovať a to : riadok na ktorom sa uskutočnila zmena, oblasť daná zmenou vo viacriadkových komentároch a oblasť daná zmenou v otvorenosti XML tagu (posledné z dvoch oblasti môžu byť prázdne). Výsledná časť textu určená na preformátovanie je daná ich zjednotením. Preformátuje sa však iba jej prienik s viditeľnou časťou. Zvyšným riadkom sa nastaví príznak o neaktuálnosti zvýraznenia.

- Pri zmenách na viacerých riadkoch – všetky zmeny ide popísať pomocou dvoch základných operácií a to :
  1. zmazanie textu.
  2. pridanie textu.

  Zmeny, ktoré môžu nastať :
  1. Zmazanie textu (klávesy Delete, Backspace, operácia Cut) – ide o uplatnenie 1 operácie.
  2. Vloženie znaku (ľubovoľná alfanumerická klávesa vrátane klávesy Enter) – je zložením operácií 1 a 2. Najprv totiž dochádza k zmazaniu označeného textu a potom k vloženiu znaku klávesy na jeho miesto.
  3. Vloženie textu (operácia Paste) – podobná situácia. Najprv dochádza k zmazaniu označeného textu a potom k vloženiu nového na jeho miesto.
  4. Vloženie šablóny – ide o uplatnenie 2 operácie.

  Popis základných operácií

  Zmazanie textu – najprv dôjde k zisteniu, či sa v označenom texte, ktorý má byť zmazaný nachádza začiatok alebo koniec viacriadkového komentára. Poznačia sa všetky začiatky a konce. Pre každý z nich sa určí aká časť textu by musela byť preformátovaná ak by bol zmazaný. Označme p_i prvý a r_i posledný riadok i-tej časti. Potom najmenšia súvislá oblasť ktorú musíme preformátovať je ohraničená riadkami $\min\{p_i \mid i = 1 \ldots \text{počet častí}\}$ a $\max\{r_i \mid i = 1 \ldots \text{počet častí}\}$. Ale keďže po zmazaní textu dôjde k posunu riadkov, tak za prvý riadok oblasti berieme riadok na ktorom sa nachádza kurzor po zmazaní a od posledného musíme odpočítať počet zmazaných riadkov. Nakoniec je nutné zvažovať (v prípade že ide o dokument typu XML) či zmazaním riadkov nedôjde k narušeniu otvorenia/zatvorenia nejakého tagu. Ak áno, tak v tom prípade je nutné do oblasti určenej na preformátovanie zahrnúť aj časť textu po najbližší začiatok alebo koniec nejakého tagu spôsobom popísaným vyššie.

príklad

```
1: <head>
2:     <title></title>
3: </head>
4: <body
5:    onload = ”inicializacia()”
6: >
7:
8: </body>
```

po zmazaní označeného textu dôjde k narušeniu otvorenosti tagu body. Je nutné preformátovanie po najbližší < alebo >. V tomto prípade je nutné preformátovanie po riadok 6 (vrátane). Teda dostávame.

```
1: <head>
2:     <title>body
3:    onload = ”inicializacia()”
4: >
5:
6: </body>
```

podobne narušenie zatvorenosti XML tagu.

Z takto získanej oblasti sformátujeme iba jej viditeľnú časť a zvyšným riadkom nastavíme príznak o nesformátovaní.

Pridanie textu – prebieha podobne len s tým rozdielom, že hľadanie začiatkov, koncov viacriadkových komentárov a zistenie narušenia otvorenosti/zatvorenosti XML tagov sa deje na riadkoch na ktorých sa nachádza vložený text.

Spojenie základných operácií – pri spojený samozrejme neformátujeme žiadnu časť dvakrát, ale urobíme zjednotenie častí z obidvoch úprav v tom zmysle, že vezmeme oblasť ohraničenú minimom z prvých riadkom a maximom z posledných riadkov častí určených na preformátovanie z použitých operácií.

Spracovanie viacriadkových komentárov a zátvoriek

Viacriadkový komentár

Pre spracovanie viacriadkových komentárov je dôležité si pamätať riadky na ktorých sa nachádza buď otváracia alebo zatváracia časť komentára. Riadok obsahujúci komentár je charakterizovaný stavom.

Rozlišujeme 2 stavy a to :

- komentár na konci riadka je otvorený – teda riadky po ním sú v ňom obsiahnuté.
- komentár na konci riadka je zatvorený – teda na danom riadku komentár končí.

príklad

Komentár je na konci riadka otvorený

```
for (int i = 0; i < 9; i++) /* komentar
```

```
/* komentar na jednom riadku */ break; /* komentar
```

Komentár je na konci riadka zatvorený

```
komentar */ if (s.GetLength() > 0){
```

```
/* komentar na jednom riadku */ nadradeny komentar */
```

Ak dôjde k zmene stavu na riadku, ktorý obsahuje komentár, resp. dôjde k jeho odstráneniu v dôsledku toho, že už nenesie žiadnu informáciu (nastáva v prípade zmazania všetkých komentárov na riadku). Je nutné preformátovanie po najbližší nižší riadok obsahujúci začiatok alebo koniec komentára (teda má nastavený nejaký stav), ktorý v dôsledku zmeny, ktorá vyvolala preformátovanie svoj stav nezmenil, ani nestratil. Ak sa taký riadok nenájde musí sa preformátovať do konca dokumentu.

príklad

```
1: while (i > 0){ /*
2:     if (i == 10) break;
3:     else {
4:           i++;
5:     }
6: }
7: if (i == 0) return;*/
```

V tomto prípade po zmazaní komentára na prvom riadku, je nutné preformátovať po riadok 7, pretože ten má nastavený stav (komentár je na konci riadka zatvorený) a po zmazaní komentára na riadku 1 sa tento stav nezmení.

```
1: while (i > 0){ /*
2:     if (i == 10) break;
3:     else {
4:           i++;
5:     }
6: }
7: string s = " */ neukonceny retazec /*
8: for (int i = 0; i < s.GetLengt(); i++){
9: }*/
```

V tomto prípade po zmazaní komentára na prvom riadku je nutné preformátovanie po riadok 9, pretože nastane následovná situácia.

```
1: while (i > 0){
2:     if (i == 10) break;
3:     else {
4:           i++;
5:     }
6: }
7: string s = " */ neukonceny retazec /*
8: for (int i = 0; i < s.GetLengt(); i++){
9: }*/
```

Odstránenie komentára spôsobilo zmenu stavu (resp. jeho stratu) na riadku 7. Predtým bol koniec riadka v komentári, teraz riadok nemá nastavený žiadny stav (pretože neobsahuje žiadny komentár). Ďalším riadkom ktorý mal nastavený

nejaký stav bol riadok 9. Koniec tohto riadka predtým v komentári nebol, po úprave prvého riadku sa stav nezmenil.

Akým spôsobom prebieha preformátovanie je popísané vyššie. V tejto časti je nutné predbehnúť a povedať, že editor si ukladá o každom riadku informácie, ktoré mu umožňujú zabezpečovať funkcie ako automatické doplňovanie zátvoriek a uplatňovanie užívateľom nadefinovanej indentácie. Tieto informácie budú v texte objasnené podrobnejšie, podstatné je že ich musí mať editor neustále k dispozíciu o každom riadku. Ak si predstavíme situáciu, že máme na začiatku dokumentu umiestnený viacriadkový komentár a zmazali by sme ho, tak by bolo nutné rozanalyzovať celý text, ktorý bol jeho súčasťou a aktualizovať informácie o riadkoch. Tu už nie je možné použiť trik ako pri zvýrazňovaní syntaxie, kedy bola formátovaná iba viditeľná časť obrazovky, pretože informácie sú potrebné neustále a o každom riadku. Na druhej strane analyzovať text po každom zmazaní viacriadkového komentára hlavne ak je umiestnený na začiatku obsiahleho kódu nie je právne najefektívnejšie. Tento problém rieši editor nasledovne : informácie o riadkoch sú aktualizované iba pri ich zmene alebo pridaní. Čo sa týka zmazania viacriadkového komentára informácie na riadkoch v ňom obsiahnutých sa nezmenili iba sa stali dostupnými.

príklad

```
1: while (i > 0){ /*
2:     if (i == 10) break;
3:     else {
4:           i++;
5:     }
6: }
7: if (i == 0) return;*/
```

Zmazanie komentára na prvom riadku nemá vplyv na informácie obsiahnuté na riadkoch 2-6. Jediné čo sa zmenilo je ich dostupnosť. Kým boli súčasťou komentára editor ich ignoroval avšak po jeho odstránení s nimi môže počítať.

Jediné, čo je nutné pri odstránení komentára zanalyzovať sú riadky na ktorých leží jeho začiatok a koniec. V uvedenom príklade je to prvý a posledný riadok. U prvého riadka je to preto, že na ňom nastala zmena u posledného preto, že koniec komentára už neplní svoju funkciu. Riadkom v rámci komentára stačí nastaviť príznak o tom, že informácie na ich riadkoch sú dostupné a môžu byť použité časťami programu, ktoré ich vyžadujú. Pri pridaní komentára je situácia podobná. Riadkom obsiahnutým v komentári je nastavený príznak o ich nedostupnosti (okrem riadkov obsahujúcich začiatok alebo koniec viacriadkového komentára). Ak vezmeme v úvahu ako prebieha zvýrazňovanie, tak sa celková réžia súvisiaca so spracovaním komentára znížila na minimum.

Zátvorky

Uvedené postupy sa aplikujú, pre všetky druhy zátvoriek rovnako.
Doplňovanie je možné aplikovať na :

- obyčajné zátvorky (,)
- hranaté zátvorky [,]
- ohraničenia XML tagov <,> (v tomto prípade nejde o zátvorky v pravom slova zmysle, boli sem zadelené z pohľadu funkcionality)

Pre aplikovanie funkcie doplňovania zátvoriek je nutné najprv implementovať vyhľadávanie ich párov. Samotné doplňovanie sa uskutoční v prípade nenájdenia páru. Pre spracovanie zátvoriek je nutné nejakým spôsobom zaznamenávať ich výskyt v texte. To sa deje na úrovni riadkov. Pre každý riadok sa ukladá počet nespárovaných zatváracích zátvoriek vychádzajúcich z jeho začiatku a počet nespárovaných otváracích zátvoriek vychádzajúcich z jeho konca. Nespárovanou zátvorkou sa myslí, že nemá pár na danom riadku, mimo neho ho samozrejme mať môže. Otváracou zátvorkou sa myslí (, [alebo <. Zatváracou),] alebo >. Pre každý typ zátvoriek sa ukladajú informácie osobitne.

príklad

```
1: if (
2:     ((a > b)&&(a < c)||
3:      (a < b)&&(a > c))||
4:     (b == c)
5: )
```

po označení spárovaných zátvoriek

```
1: if (
2:     ((a > b)&&(a < c)||
3:      (a < b)&&(a > c))||
4:     (b == c)
5: )
```

uložené informácie na riadkoch (počet nespár. otváracích, počet nespár. zatváracích)
1 riadok – (1, 0)
2 riadok – (1, 0)
3 riadok – (0, 1)
4 riadok – (0, 0)
5 riadok – (0, 1)

Vyhľadávanie páru nastáva po napísaní zátvorky. Po napísaní otváracej sa hľadá zatváracia a naopak. Vyhľadávanie páru je nasledovné : Najprv sa zistí, či zátvorka nemá pár na danom riadku. Ak nie, tak nastáva hľadanie na riadkoch vyšších alebo nižších podľa toho, či je zátvorka zatváracia alebo otváracia. Predpokladajme, že chceme nájsť pár pre zatváraciu zátvorku. Pre otváraciu je postup rovnaký až na smer hľadania. Najprv zistíme počet (označme ho N) nespárovaných zatváracích zátvoriek na danom riadku vychádzajúcich zo začiatku riadka, ktoré sa nachádzajú pred danom zátvorkou ku ktorej hľadáme pár, pretože tie musia byť spárované najskôr. Teda vlastne hľadáme páry k N+1 zátvorkám. Teraz sa postupne prechádzajú riadky smerom k začiatku dokumentu a na základe počtov (označme ako M) nespárovaných otváracích zátvoriek sa určí, či sa na danom riadku nachádza pár pre hľadanú zátvorku. Ak je ich počet väčší rovný ako N+1, tak potom môžeme každej nespárovanej zatváracej zátvorke (z N+1 zátvoriek ku ktorým hľadáme pár) priradiť jednu z M otváracích a párová k nami hľadanej bude M-N tá v poradí. Ak je však ich počet menší, potom môžeme na danom riadku spárovať iba M zátvoriek a teda nám ostáva (N+1)-M zátvoriek, ku ktorým musíme nájsť páry na vyšších riadkoch. K nim však ešte musíme pripočítať nespárované zatváracie zátvorky z daného riadka (označme K), pretože tie musia byť spárované pred nimi. Teda nakoniec počet zátvoriek ku ktorým hľadáme páry na vyšších riadkoch je (N+1)-M+K. Takto postupujeme kým nenájdeme vhodný riadok, alebo kým nenatrafíme na prvý. V prípade, že ani on nie je vyhovujúci zátvorka nemá pár. Keďže v pri hľadaní nedochádza k žiadnej analýze textu, tak je maximálne efektívne.

Informácie na riadkoch

Pre zaistenie funkcií ako je zvýrazňovanie zátvoriek po nájdený páru, vyhodnotenie a uplatnenie indentačných pravidiel, spracovanie viacriadkových komentárov, ... je nutné udržiavať o riadkoch isté informácie.

Informácie, ktoré sa uchovávajú sú.

- Príznak o sformátovaní - ukladá sa pre každý riadok. Označuje, či daný riadok je sformátovaný alebo nie.
- Príznak o dostupnosti informácii – ukladá sa pre každý riadok. Označuje, či sú informácie na riadku dostupné, teda či sa riadok nachádza mimo viacriadkového komentára a teda môžu byť využívané napr. pri uplatňovaní indentačných pravidiel.
- Informácie o nespárovaných zátvorkách – ukladajú sa pre každý riadok. Informácie sú potrebné pre zabezpečenie funkcií ako automatické doplňovanie zátvoriek a vyhľadávanie páru.
- Informácia o otvorenosti XML tagu – ukladá sa pre každý riadok v prípade, že dokument je typu XML. Označuje, či nejaký XML tag pokračuje na nižších riadkoch, teda či je zapísaný na viac ako jednom riadku.
- Stav viacriadkového komentára – ukladá sa na riadkoch, ktoré obsahujú jeho začiatok alebo koniec.
- Tokeny rozpoznaných kľúčových slov – ukladajú sa pre každý riadok. Ide o zoznam tokenov kľúčových slov na danom riadku, ktoré sú obsiahnuté v definíciách indentačných pravidiel a teda sú potrebné pre analýzu ich uplatnenia.

Formátovanie riadka je vždy spojené s aktualizovaním informácii o ňom. Pri formátovaní textu na viacerých riadkoch sa informácie o riadkoch implicitne neaktualizujú, pretože v niektorých prípadov je to zbytočné až nežiadúce kvôli strate efektivity (napr. spracovanie viacriadkových komentárov).

Aktualizácia informácii nastáva :

- Pri otvorení alebo vytvorení nového dokumentu – dochádza k analyzovaniu celého dokumentu a aktualizácií všetkých typov informácií.
- Pri zmene na jednom riadku – dochádza k aktualizácií všetkých typov informácií na danom riadku.
- Pri zmene na viacerých riadkoch - dochádza k aktualizácií všetkých typov informácií na riadkoch ktorých sa zmena dotkla, resp. riadkoch ktoré boli pridané.
- Pri zrušení alebo zavedení viacriadkového komentára – dochádza k aktualizácii príznakov o dostupnosti informácií na riadkoch obsiahnutých v komentári.
- Pri vložení šablóny – dochádza k aktualizácií všetkých typov informácií na riadkoch na ktorých sa nachádza vložená šablóna.

4.3 Vloženie šablóny

Nastáva po jej vybratí z kontextového menu, alebo ak dôjde k jej automatickému vloženiu. Šablóna sa vkladá ako obyčajný nesformátovaný text (vkladá sa iba časť obsahu šablóny, ktorá nebola užívateľom zadaná). Po jej vložení sa na všetky riadky, ktoré ju obsahujú uplatní najprv zvýraznenie syntaxie (tým sa z nej stáva súčasť

zdrojového kódu) a potom indentačné pravidlá, ktoré ju správne začlenia do kódu. Nakoniec sa nastaví kurzor na počiatočnú pozíciu šablóny.

príklad

```
1: if (!uspech) break;
2: fo
```

Majme definovanú obyčajnú šablónu s obsahom napr. for (int i = 0; i < ; i++) a s počiatočnou pozíciou kurzora 20. Po napísaní znaku "o" nastane porovnanie obsahu šablóny s práve editovaným textom, ktorým je "fo". Porovnanie uspeje a teda šablónu je možné použiť. Zobrazí sa na výber z kontextového menu. Po jej vložení nastane nasledovná situácia.

```
1: if (!uspech) break;
2: for (int i = 0; i <  ; i++)
```

Vložil sa zvyšok obsahu šablóny a kurzor sa nastavil na počiatočnú pozíciu.

```
1: <
2:
3: bod
```

Majme definovanú XML šablónu s párovým tagom body. Po napísaní znaku "d" nastane vyhľadanie začiatku tagu rovnakým algoritmom aký je použitý pri párovaní zátvoriek. Ten je nájdený na prvom riadku. Potom vezmeme text ležiaci od jeho začiatku po pozíciu kurzora z ktorého odstránime počiatočné biele znaky. Výsledkom je text "bod". Nastáva zistenie, či je začiatkom tagu niektorej definovanej XML šablóny. Operácia uspeje pre XML šablónu body. Zobrazí sa na výber z kontextového menu. Po jej vložení nastane nasledovná situácia.

```
1: <
2:
3: body></body>
```

Vložil sa zvyšok obsahu šablóny. Výhoda z použitia XML šablóny, je že šablónu je možné použiť iba v otvorenom XML tagu a taktiež je jej použitie flexibilnejšie, pretože medzi začiatkom tagu < a jeho obsahom môže byť ľubovoľné množstvo bielych znakov. Rozdiel je ešte markantnejší pri napísaní jej konca, teda znaku ">".

4.4 Uplatnenie indentačného pravidla

Editor sa snaží uplatniť niektoré zo zadaných pravidiel v týchto prípadoch

1. pri vložení nového riadka, teda stlačení klávesy Enter.
2. pri napísaní kľúčového slova na začiatok riadka.
3. pri vložení šablóny.
4. ak o to užívateľ požiada.

1. a 2. pri vložení nového riadka sa skúšajú uplatniť pravidlá, ktoré v indentačnej časti svojej definície nemajú na začiatku kľúčové slovo. Pri napísaní kľúčového slova na začiatku riadka sa skúšajú uplatniť pravidlá, ktoré v indentačnej časti svojej definície majú na začiatku dané kľúčové slovo.
3. ak užívateľ vloží šablónu, tak sa indentačné pravidlá skúšajú uplatniť na všetky riadky, na ktorých sa nachádza okrem prvého, pretože sa predpokladá, že užívateľ chce sám určiť pozíciu jej začiatku.
4. ak užívateľ označí text a požiada o jeho indentáciu, tak sa indentačné pravidlá skúšajú uplatniť na všetky riadky obsiahnuté v označení.

Ak je text editovaný v rámci viacriadkového komentára, tak pri vložený nového riadka je uplatnená indentácia riadka predchádzajúceho (auto indentation). Teda na jeho začiatok je vložené odsadenie predchádzajúceho riadka.

Samotné uplatňovanie pravidiel prebieha nasledovne. Najprv sa vyberú pravidlá, ktoré sa môžu z daného kontextu uplatniť (rozhoduje indentačná časť) a nastáva proces nasadenia vzoru pravidla na text od editovaného riadka smerom k začiatku dokumentu. Každý zo špeciálnych symbolov sa snaží pohltiť čo najmenej textu. Pri analýze sa ignorujú všetky medzery, tabulátory a prázdne riadky. Ak pri vyhľadávaní vzoru dôjde k stavu, že nie je nájdený celý vzor a zároveň nie je možné pokračovať ďalej, tak je nutné sa vrátiť k poslednému špeciálnemu symbolu a nechať ho pohltiť viac textu ak je to možné a pokračovať ďalej, ak to možné nie je tak daný vzor nie je možné nájsť a pravidlo sa nemôže uplatniť.

príklad

definície pravidiel

```
BLOK { }
PRAVIDLO_BEGIN
        if
        (
        ...
        )
        .B.
PRAVIDLO_INDENTACIA
        ODKLUCOVEHOSLOVA 0 X
PRAVIDLO_END
```

vstupný text

```
1: if (a == b){
2: }
3: a = (1 + 2) * 3;
4:
```

Po stlačení Enter na riadku 3 sa dostávame na riadok 4 a začína sa vyhľadávať vzor. Na riadku 3 je nájdená časť vzoru (…) .B. symbol … pohltil text "1 + 2" a symbol .B. pohltil text "* 3" avšak ďalej už pokračovať nejde keďže po (sa očakáva if a namiesto neho tam je =. Teda sa musíme pri vyhľadávaní vzoru vrátiť naspäť k symbolu ... a nechať ho pohltiť viac textu až kým nenájde (. Symbol … teda ešte pohltí text "a = (" ale ďalej už pokračovať nemôže pretože natrafil na začiatok bloku {. Je nutné sa vrátiť k symbolu .B. a nechať ho pohltiť viac textu až kým nenatrafí na). To sa podarí na riadku 1 a nasledujúci text odpovedá zvyšku vzoru, teda pravidlo ide uplatniť. Vzor sa našiel na riadkoch 1 až 3.

Komplikovanejšia situácia nastáva ak má pravidlo nastavenú voľbu R. Táto voľba má zmysel iba ak sa má pravidlo uplatniť po napísaní kľúčového slova. Vyhľadávanie prebieha rovnako avšak s tým rozdielom, že po nájdení pravidla sa musí overiť, či sa nájdený vzor nespája s už uplatneným pravidlom tej istej definície. Ak nie, pravidlo sa uplatní, ak áno pokračuje sa s vyhľadávaním vzoru na vyšších riadkoch podľa popisu vyššie. Pre účel overenia je nutné pri vyhľadávaní vzoru poznamenať pozície všetkých výskytov slova ktoré vyvolalo uplatňovanie pravidla. Z týchto pozícií sa potom skúša nasadiť pravidlo rovnakej definície. V prípade úspechu je nutné overiť, či sa prvé slovo nájdeného vzoru zhoduje s prvým slovom nájdeného vzoru pôvodného pravidla, ktoré

sa snažím uplatniť. Ak áno, tak došlo k situácii, kedy sa nájdený vzor viaže s už skôr uplatneným pravidlom rovnakej definície.

príklad

definície pravidiel

```
BLOK { }
PRAVIDLO_BEGIN R
        if
        .B.
PRAVIDLO_INDENTACIA
        else ODKLUCOVEHOSLOVA 0 X
PRAVIDLO_END
```

vstupný text

```
1: if (i > 10){
2: }
3: else{
4: }
5: else
```

Po napísaní else na 5 riadku sa síce vzor pravidla nájde, ale pravidlo sa neuplatní, pretože už uplatnené bolo a to počínajúc riadkom 3. Ak by však bola situácia trochu iná a to

vstupný text

```
1: if (b == true)
2: if (i > 10){
3: }
4: else{
5: }
6: else
```

Teraz po napísaní else na 6 riadku by došlo k nájdeniu vzoru pravidla na riadkoch 2 až 5 avšak ten sa už viaže s else na riadku 4 a teda sa musí vo vyhľadávaní vzoru pokračovať vyššie. Ďalšie nájdenie vzoru je na riadkoch 1 až 5 a tento vzor sa už neviaže k žiadnemu inému else. Teda dané pravidlo sa uplatní.

4.5 Funkcie editora

Scrollovanie

Pri scrollovaní dochádzalo k nasledujúcemu problému. Sformátovanie viditeľnej časti, odkrytej pomocou scrollbaru, mohlo vyvolať zmenu veľkosti niektorých riadkov (kvôli nastaveniu veľkosti písma istých elementov) dôsledkom čoho došlo k aktualizácii minimálnej a maximálnej hodnoty ktorú môže nadobúdať posuvník a tým pádom aj k aktualizácii textu vo viditeľnej časti obrazovky, tak aby odpovedala jeho pozícii. Teda mohlo dôjsť k odkrytiu nesformátovaného textu. Kvôli danému problému bolo nutné zabezpečiť aby mali všetky kľúčové slová a špeciálne elementy rovnakú veľkosť písma. Tá sa nastavuje iba pre obyčajný text a je spoločná pre všetky zobrazované elementy. Niektoré z existujúcich editorov túto situáciu riešia tým, že nastavia fixne výšku všetkých riadkov podľa elementu s najväčšou veľkosťou písma. Dôsledkom tohto prístupu je však to, že v prípade ak sa na riadku nevyskytuje daný element tak

dochádza k zbytočnému plytvaniu miestom a k nie práve najestetickejšiemu vzhľadu zdrojového kódu.

Ďalším problémom je, že po zmene viditeľnej časti scrollovaním sa musí užívateľovi text v nej obsiahnutý prezentovať v sformátovanej podobe. Z toho plynie, že pri scrollovaní je nutné zistiť výslednú pozíciu posuvníka a teda časť dokumentu ktorá sa stane viditeľnou pred tým ako sa zobrazí užívateľovi, pretože je ju nutné najskôr sformátovať. Z tohto dôvodu bolo nutné implementovať vlastnú obsluhu scrollovania.

Print preview

Print preview je dialógové okno slúžiace na zobrazenie dokumentu pred samotným tlačením. Keďže v editore sa formátuje iba text vo viditeľnej časti obrazovky, tak po vyvolaní daného dialógu často nastáva situácia, že je zdrojový kód sformátovaný nekompletne. Takýto dokument samozrejme nemôže slúžiť na požadovaný účel. Riešením daného problém by bolo preformátovanie celého dokumentu pred samotným zavolaním dialógu, avšak pri veľkých dokumentoch by táto operácia vyžadovala nemalý čas. Preto som sa rozhodol daný problém vyriešiť nasledovne. Dialóg zabezpečujúci print preview poskytuje náhľady jednotlivých strán, pričom je možné spracovať udalosť, ktorá je vyvolaná pri zmene strany. V tejto udalosti najprv zistím, ktorá strana sa má zobraziť, jej prvý riadok a veľkosť (takisto v riadkoch). Text určený jej prvým riadkom a veľkosťou strany musím sformátovať pred samotným zobrazením strany v dialógu. Tým sa časová náročnosť celej operácie rozloží tak, že si jej užívať nebude vedomý.

Undo

Použitá MFC komponenta má v sebe zabudovanú podporu undo operácie, teda operácie, ktorá je schopná vrátiť dokument do stavu pred poslednou zmenou. Avšak tú je možné použiť iba pri editovaní obyčajného textu. Ak by pri jej vyvolaní došlo napríklad k zrušeniu začiatku viacriadkového komentára bolo by nutné preformátovanie textu po jeho koniec a to mi už daná operácia zabezpečiť nevie.

Undo som teda v editore implementoval nasledovne. Pri jeho vyvolaní ignorujem jeho štandardnú obsluhu komponentou a použijem vlastnú. Na začiatok je dôležité povedať aké informácie si je nutné uložiť o každej zmene aby ju bolo možné vrátiť späť. Každá zmena je reprezentovaná trojicou

(zmazaný text, vložený text, pozícia kurzora po zmene).

Zmazaný text
je text, ktorý sa nachádza v označení pred zmenou a teda bude zmazaný.
Vložený text
je text, ktorý bude pri zmene vložený. Pri operácii delete bude tento text samozrejme prázdny.
Pozícia kurzora po zmene
ide o pozíciu kurzora vzhľadom k začiatku dokumentu.

príklad

Stlačenie alfanumerickej klávesy 0

```
pred : if (uspech == false) break;
po   : if (uspech == 0) break;
```

Trojica reprezentujúca zmenu je ("false", "0", 15). Samozrejme iba v prípade ak sa riadok vyskytuje na začiatku dokumentu.

Samotná operácia undo prebieha takto. Najprv dôjde k nastaveniu kurzora na pozíciu po zmene. Na tejto pozícii končí text, ktorý bol vložený. Označí sa a následne nahradí zmazaným textom. Tým sa dokument ocitol v stave pred zmenou. To je avšak len funkcionalita, ktorú by vykonala aj samotná komponenta. Na mazanom a vkladanom texte, je nutné uplatniť základné operácie popísané v časti vývojovej dokumentácie týkajúcej sa zvýrazňovania, pri ktorých dochádza k analýze zmeny viacriadkového komentára a otvorenosti XML elementov a k následnému preformátovaniu ak je potrebné.

Undo bolo nutné implementovať pri nasledujúcich operáciách :

- stlačenie alfanumerickej klávesy vrátane enter
- zmazanie, vloženie a vystrihnutie textu
- vloženie šablóny
- uplatnenie indentácie
- doplnenie zátvorky

Editor poskytuje undo operáciu do neobmedzenej hĺbky. Na ňom je postavené zistenie modifikácie dokumentu. Ak je zásobník pre undo prázdny, dokument je v rovnakom stave ako pred jeho otvorením alebo vytvorením a teda pri jeho zatváraní nebude zobrazovaný dialóg na uloženie vykonaných zmien.

5 Porovnanie s inými programami

V tejto kapitola sa pokúsim o porovnanie s programami rovnakého alebo podobného určenia. Pričom sa zameriam hlavne na funkcie priamo súvisiace s editovaním kódu akými sú zvýrazňovanie syntaxie, podpora indentácie, šablón a pod. Na podrobné porovnanie by bolo nutné uviesť desiatky programov, čo samozrejme nie je možné, preto som sa rozhodol pre 4 vhodných reprezentantov. Notepad som zvolil ako zástupcu editora, ktorý nie je priamo určený na editovanie zdrojového kódu, ale je možné ho na daný účel použiť. Vim ako zástupca legendárneho editora oplývajúceho nesmiernou funkčnosťou. Strednú triedu editorov reprezentujú Notepad++ a PSPad (zrejme najúspešnejší tuzemský editor).

Notepad

Program je štandardnou súčasťou operačného systému Windows. Slúži na editovanie obyčajného textu, teda je možné ho použiť i na editovanie zdrojového kódu. Program dokáže editovať iba jeden súbor zároveň. Nepodporuje zvýrazňovanie syntaxie, akúkoľvek indentáciu, doplňovanie zátvoriek ani používanie šablón. V programe je možné zmeniť formátovanie písma, ale iba v rámci celého dokumentu. Ďalej program poskytuje bežné operácie akými sú kopírovanie, vystrihnutie, vloženie textu, jeho vyhľadávanie a nahradzovanie iným. Taktiež je možné výsledný dokument vytlačiť, ale nie je možné jeho zobrazenie pre samotnou tlačou (print preview). Editor teda nie je vhodné používať na editovanie zdrojových kódov. Uviedol som ho sem hlavne preto, aby som zvýraznil rozdiel medzi editormi obyčajného textu a programátorskými editormi.

Notepad++

Ide o freewarový program postavený na komponente Scintilla. Umožňuje editovanie viacerých dokumentov súčasne. Obsahuje štandardné funkcie ako kopírovanie, vkladanie textu, vrátenie uskutočnených zmien (undo) a pod. Dokáže vyhľadávať a nahradzovať text vyjadrený pomocou regulárnych výrazov. Čo sa týka syntaktických funkcií, tak obsahuje doplňovanie zátvoriek a vyhľadávanie odpovedajúceho páru, preddefinované zvýrazňovanie pre najpoužívanejšie programovacie jazyky, pričom je možné nadefinovať zvýrazňovanie pre nový jazyk. V editore je ďalej možné vkladať šablóny, definovať vlastné pre konkrétny jazyk a predať práve editovaný dokument na spracovanie inému programu. Ďalšou užitočnou funkciou je code folding, teda skrytie časti zdrojového kódu, ktorá odpovedá napr. obsahu funkcie, cyklu alebo viacriadkového komentára. U užívateľsky definovaných jazykoch je možné nadefinovať začiatok a koniec bloku (prostredníctvom zadania kľúčových slov) na ktorý je možné následne code folding použiť. Najväčšou nevýhodou daného editora je, že neposkytuje indentáciu ani na úrovni blokov. Používa tzv. auto indentation, teda indentácia sa pri vložení nového riadka kopíruje z predchádzajúceho.

PSPad

Ide o freewarový program umožňujúci editovanie viacerých súborov súčasne. Podporuje štandardné funkcie ako kopírovanie, vkladanie textu, vrátenie zmien, vyhľadávanie a nahradzovanie textu a to i vyjadreného pomocou regulárneho výrazu. Umožňuje doplňovanie uzatváracích zátvoriek a vyhľadávanie ich párov. V editore je zavedená podpora pre mnohé jazyky, čo sa týka jednak zvýrazňovania ich syntaxie

a ďalších funkcií akými sú napr. prieskumník kódu. Editor umožňuje nadefinovať zvýrazňovanie pre nový jazyk, avšak v ňom je možné definovať iba skupiny kľúčových slov, ktoré stoja osobitne, resp. sú obklopené oddeľovačmi (napr. operátormi). Oddeľovače sú umiestnené vo vopred danej skupine, ktorá sa nedá meniť a teda užívateľ nemôže nadefinovať jazyk s inou množinou oddeľovačov alebo nadefinovať aby sa nejaká ich podmnožina zobrazovala inak. Editor ďalej umožňuje používanie a definovanie vlastných šablón. Jeho nevýhodou je opäť indentácia, pretože nepodporuje ani smart a ani len blokovú indentáciu. Tá je riešená tak, že sa odsadenie kopíruje z predchádzajúceho riadka. Naproti tomu editor disponuje množstvom rozličných funkcií akými sú napr. definovanie makier, podpora pre rôzne kódovania súborov, HEXA editor, export do RTF, HTML, XHTML, TeX, kontrola pravopisu, otvorenie dokumentu v užívateľom zadaným programom a mnohé ďalšie.

Vim

Je zrejme najznámejší programátorský editor. Pôvodne určený pre systém UNIX ale v dnešnej dobe existujú jeho verzie i pre ďalšie operačné systémy. Nejde o klasický editor ako vyššie uvedené. Uvádzam ho tu predovšetkým preto aby som poukázal na jeho funkcionalitu. Editor poskytuje všetky štandardné funkcie ako kopírovanie, vkladanie textu a pod. Pri nahradzovaní a vyhľadávaní textu je možné použiť regulárne výrazy. Je možné nadefinovať vlastné makrá, zvýrazňovanie syntaxie, pravidlá, ktorými sa má riadiť indentácia pre daný programovací jazyk. Podporuje používanie a vytváranie šablón, code folding, kontrolu pravopisu, porovnávanie súborov a množstvo ďalších funkcií. Jeho najväčšou nevýhodou je nie veľmi užívateľský príjemné prostredie a zložité ovládanie, ktoré je v dnešnej dobe neštandardné. Veľká časť editačných úprav je realizovaná prostredníctvom príkazov. Nastavovanie zvýrazňovania syntaxie, alebo indentačných pravidiel je možné jedine ich naeditovaním v príslušných konfiguračných súboroch pomocou určeného spôsobu zápisu. Dané nevýhody sa snaží odstrániť jeho grafická verzia. No i napriek tomu pomerne komplikované ovládanie odrádza mnohých nie len začínajúcich programátorov od jeho používania.

6 Popis priloženého CD

Súčasťou bakalárskej práce je priložené CD, ktoré obsahuje výsledný editor a text bakalárskej práce v elektronickej podobe.

Obsah priloženého CD :

- adresár *Zdrojový kód* – obsahuje zdrojové kódy programu, teda projekt pre Microsoft Visual Studio 2005.
- adresár *Program* – obsahuje výsledný editor.
- adresár *Text* – obsahuje text bakalárskej práce vo formáte pdf a doc.
- README.txt – obsahuje popis CD a inštalácie programu.

7 Záver

Cieľom tejto práce bolo implementovať editor programátorského štýlu pre ľubovoľný programovací jazyk. Umožniť funkcie ako zvýrazňovanie syntaxie, doplňovanie zátvoriek, kľúčových slov, XML elementov a nakoniec zaviesť podporu pre užívateľom zadanú indentáciu pomocou vhodne zvolených syntaktických pravidiel. Danému zadaniu nie je možné vyhovieť úplne kvôli mnohým rozlišnostiam, ktoré jednotlivé jazyky obsahujú. Samotné pridanie podpory programovacieho jazyka do editoru sa realizuje prostredníctvom vytvorenia štýlu. Ten je nositeľom všetkých nastavení, ktoré charakterizujú daný jazyk a zabezpečujú jeho správne editovanie. Najväčšiu rozdielnosť je možné badať pri jazykoch odvodených od XML. Pre tento účel bola pridaná možnosť nastavenia na akom type dokumentu sa budú spomínané funkcie vykonávať. Pri dokumente so štruktúrou XML má zmysel realizovať zvýrazňovanie syntaxie alebo doplňovanie kľúčových slov iba v rámci jednotlivých tagov, pretože všetko mimo nich je interpretované ako obyčajný text.

Zvýrazňovanie syntaxie je rozdelené na dve časti a to na zvýrazňovanie kľúčových slov a špeciálnych elementov akými sú komentáre, reťazce uzavreté úvodzovkami a apostrofmi, identifikátory, čísla a obyčajný text. Zvýrazňovanie kľúčových slov sa realizuje prostredníctvom vytvárania skupín, ktoré sú nositeľmi formátovacích konfigurácií. Nastavenie zobrazenia pre dané kľúčové slovo je uskutočnené jeho pridaním do príslušnej skupiny. Pri identifikátoroch je možné zadať zoznam znakov, ktoré sú povolené pomimo číslic a písmen.

Doplňovanie kľúčových slov a XML elementov je uskutočnené prostredníctvom šablón. Tie typicky obsahujú syntaktickú konštrukciu, ktorá môže byť vložená v určitom kontexte. Šablóny sú rozdelené do dvoch typov a to na obyčajné šablóny a XML šablóny, ktoré sa používajú pri doplňovaní XML elementov.

Indentácia je realizovaná pomocou pravidiel zadaných užívateľom. Každé pravidlo obsahuje vzor popisujúci situáciu v ktorej má byť uplatnené. Jeho súčasťou je takisto aj zadanie indentácie, ktorá bude použitá v prípade nájdenia vzoru. Pravidlá sa môžu uplatniť buď pri pridaní nového riadku alebo napísaní kľúčového slova na jeho začiatok. Pre uľahčenie práce pri zarovnávaní zdrojového kódu bola pridaná možnosť uplatnenia indentačných pravidiel na označenom texte.

Výsledný program podporuje editovanie viacerých dokumentov súčasne a obsahuje štandardné funkcie pre textový editor ako ukladanie, otváranie dokumentov, tlačenie a zobrazenie náhľadu pre tlačením, vyhľadávanie a nahradzovanie textu, operáciu undo neobmedzenej hĺbky, vystrihnutie, kopírovanie a vkladanie textu.

7.1 Možnosti rozšírenia

V ďalšom vývoji editora by mohlo dôjsť k vylepšeniu použitia šablón napríklad ich parametrizáciou. Teda napr. k šablóne popisujúcej for cyklus by bolo možné pridať názov riadiacej premennej a na základe toho vygenerovať dynamicky jej obsah, ktorý bude vložený. Taktiež veľmi užitočnou funkciou je code folding. Jedná sa o možnosť skrytia časti zdrojového kódu napr. tela funkcie alebo viacriadkového komentára. To umožňuje nechať zobrazený iba zdrojový kód ktorý je práve potrebný. Čo sa týka ďalších vylepšení, tak pre ne neexistuje žiadny limit, keďže táto téma nie je ani zďaleka vyčerpaná, práve naopak. O tom svedčia aj každým rokom pribúdajúce editory

podporujúce stále nové a nové funkcie. A je len otázkou času kedy sa z nich stanú štandardy, ktoré bude musieť implementovať každý editor aby uspel už v dnes nemalej konkurencii.

Literatúra

[1] M. Chytil : *Automaty a gramatiky*, SNTL Praha, 1984

[2] Teitelbaum, T., Reps, T. : *The Cornell program synthesizer: a syntax-directed programming environment,* Commun. ACM 24, 9 (Sep. 1981), 563-573. ISSN:0001-0782

[3] Zelkowits, M. V. : *A small contribution to editing with a syntax directed editor,* SIGPLAN Not. 19, 5 (May. 1984), 1-6. ISSN:0362-1340

[4] Špecifikácia formátu RTF
http://msdn.microsoft.com/en-us/library/aa140277.aspx

[5] Špecifikácia jazyka HTML
http://www.w3.org/TR/html401/

[6] Slidy k prednáške : Automaty a gramatiky
http://kti.mff.cuni.cz/~bartak/automaty/prednaska.html

[7] Slidy k prednáške : Principy překladačů
http://ulita.ms.mff.cuni.cz/pub/predn/PP/

[8] Slidy k prednáške : Doporučené postupy v programování
http://dsrg.mff.cuni.cz/teaching/nprg043/

[9] Referenčná príručka k MFC
http://msdn.microsoft.com/en-us/library/d06h2x6e(VS.80).aspx

[10] J. Prosise : *Programing Windows with MFC Second Edition*, Microsoft Press, 1999