Konstrukcja kompilatorów — przetwarzanie plików XML

1 Cel

Celem ćwiczenia jest doskonalenie umiejętności tworzenia analizatorów leksykalnych i składniowych rozbudowanych o analizę semantyczną. W trakcie realizacji ćwiczenia powinien powstać program, który sprawdzając poprawność uproszczonego pliku XML drukuje go w sposób uwzględniający zagnieżdżenie znaczników.

Ćwiczenie jest także pomyślane jako sprawdzian większej samodzielności studentów, stąd dostarczona do niego jest tylko instrukcja, a brak jest dodatkowych plików ze szkieletem programu. Plik `Makefile` można skopiować i przystosować z wcześniejszych ćwiczeń. Dostarczono pliki testowe.

2 Analiza leksykalna

Analizator leksykalny należy umieścić w pliku `x.l`.

2.1 Deklaracje

W sekcji deklaracji:

1. Należy dołączyć plik `defs.h` zawierający makrodefinicję największej dopuszczalnej długości napisu `MAXSTRLEN`.
2. Należy dołączyć plik `x.tab.h` zawierający definicje z analizatora składniowego.
3. Należy nazwać składnię identyfikatora, ponieważ identyfikator jest częścią kilku rozpoznawanych konstrukcji. Pierwszym znakiem identyfikatora może być litera, dwukropek lub znak podkreślenia; dalsze znaki mogą dodatkowo zawierać cyfry i myślnik.
4. Należy zadeklarować warunki początkowe `ST_COMMENT` (środek komentarza), `ST_PI` (środek instrukcji przetwarzania) i `ST_TAG` (środek znacznika).

2.2 Reguły

W trakcie analizy leksykalnej:

1. Usuwane są komentarze, Komentarze zaczynają się znakami „`<!--`”, a kończą znakami „`-->`” poprzedzonymi znakiem różnym od myślnika.
2. Wykrywany jest brak zamknięcia komentarza.
3. Wykrywany jest początek instrukcji przetwarzania w postaci znaków „`<?`” z następującym bezpośrednio po nich identyfikatorem. Początek instrukcji przetwarzania zwracany jest jako symbol końcowy `PI_TAG_BEG`, a jego wartością jest tekst identyfikatora.
4. Pomijane jest wnętrze instrukcji przetwarzania.
5. Wykrywany jest koniec instrukcji przetwarzania „`?>`” zwracany jako `PI_TAG_END`.
6. Wykrywany jest początek znacznika otwierającego w postaci znaku „`<`” z następującym bezpośrednio po nim identyfikatorem i zwracany jako symbol końcowy `STAG_BEG`. Wartością tego symbolu jest tekst identyfikatora.
7. Wykrywany jest początek znacznika zamykającego w postaci znaków „`</`” z następującym bezpośrednio po nich identyfikatorem i zwracany jako symbol końcowy `ETAG_BEG`. Wartością tego symbolu jest tekst identyfikatora.
8. Wnętrze znaczników jest pomijane.
9. Wykrywany jest koniec znacznika postaci „`>`” i zwracany jako symbol końcowy `TAG_END`.

10. Wykrywany jest koniec znacznika pustego postaci „/>” i zwracany jako symbol końcowy ETAG_END.
11. Napisy „<”, „>”, „&”, „'” i „&quote;” są zwracane jako symbol końcowy CHAR o wartości odpowiednio znaków „<”, „>”, „&”, apostrofu i cudzysłowu.
12. Ciągi odstępów i znaków tabulacji poziomej na początku wiersza są pomijane.
13. Znak nowego wiersza jest zwracany jako znak nowego wiersza.
14. Znak powrotu karetki jest pomijany.
15. Znaki odstępu i tabulacji poziomej są zwracane jako symbol końcowy S. Ich wartością jest dopasowany znak.
16. Pozostałe znaki są zwracane jako symbol końcowy CHAR o wartości będącej dopasowanym znakiem.

2.3 Funkcje

Funkcja yywrap() zwraca wartość 1 i wykrywa brak końca komentarza, instrukcji przetwarzania lub znacznika.

3 Analiza składniowa

Analizator składniowy powinien zostać umieszczony w pliku x.y.

3.1 Deklaracje

W sekcji deklaracji:

1. Należy dołączyć pliki nagłówkowe bibliotek stdio.h i string.h oraz plik nagłówkowy defs.h zawierający makrodefinicję stałej MAXSTRLEN.
2. Należy zadeklarować zmienną level o wartości początkowej 0 reprezentującą poziom zagnieżdżenia znaczników.
3. Należy zadeklarować zmienną pos o wartości początkowej 0 reprezentującą bieżącą kolumnę tekstu.
4. Należy zadeklarować stałe INDENT_LENGTH i LINE_WIDTH reprezentujące szerokość pojedynczego wcięcia (np. 2) i największą liczbę znaków w wierszu (np. 78).
5. Należy umieścić nagłówek funkcji indent() dokonującej wcięcia z parametrem będącym poziomem zagnieżdżenia.
6. Należy umieścić dyrektywę %union z polem znakowym s mieszczącym MAXSTRLEN+1 znaków.
7. Należy zadeklarować symbole końcowe (o wartościach znakowych): PI_TAG_BEG, PI_TAG_END, STAG_BEG, ETAG_BEG, TAG_END, ETAG_END, CHAR i S.
8. Należy zadeklarować zmienne gramatyki (symbole niekońcowe) start_tag, end_tag i word jako symbole o wartościach będących napisami.

3.2 Reguły

1. Dokument XML składa się ze wstępu i elementu.
2. Wstęp składa się ciągu instrukcji przetwarzania i znaków nowego wiersza.
3. Instrukcja przetwarzania składa się z początku instrukcji przetwarzania (PI_TAG_BEG) i końca tej instrukcji (PI_TAG_END).
4. Element składa się z pustego znacznika lub z pary elementów.

5. Pusty znacznik składa się z początku znacznika (`STAG_BEG`) i końca pustego znacznika (`ETAG_END`).
6. Para znaczników składa się ze znacznika otwierającego, zawartości i znacznika kończącego.
7. Znacznik otwierający składa się z początku znacznika (`STAG_BEG`) i końca znacznika (`TAG_END`).
8. Znacznik kończący składa się z początku znacznika kończącego (`ETAG_BEG`) i końca znacznika (`TAG_END`).
9. Zawartość jest ciągiem elementów, białych znaków (`S`), słów (ciągów znaków różnych od białych) i znaków nowego wiersza.

3.3 Funkcje

Należy zdefiniować:

1. funkcję `main()`, która wywołuje funkcję `yyparse()`,
2. funkcję `yyerror()`, która wypisuje podany komunikat i zwraca 0,
3. funkcję `indent()`, która dokonuje wcięć.

4 Zadania

Należy:

1. zrealizować analizator leksykalny,
2. zrealizować analizator składniowy sprawdzający poprawność składniową analizowanego pliku XML,
3. sprawdzać dopasowanie pary znaczników i wypisywać komunikat w przypadku braku dopasowania,
4. wypisywać znalezione znaczniki z nazwami ale z pominięciem wnętrza znaczników,
5. wypisywać znalezione instrukcje przetwarzania z nazwami ale z pomięciem wnętrza instrukcji,
6. wypisywanie znaków i instrukcji przetwarzania z wcięciami wynikającymi z poziomu zagnieżdżenia,
7. wypisywanie tekstu między znacznikami z wcięciami wynikającymi z poziomu zagnieżdżenia,
8. wypisywanie tekstu między znacznikami z wcięciami wynikającymi z poziomu zagnieżdżenia z łamaniem tekstu, tak aby nie wykraczał poza zadeklarowaną szerokość wiersza (`LINE_WIDTH`). Łamanie tekstu może być uproszczone, tzn. można przenosić do nowego wiersza całe słowa — nie jest konieczne dzielenie ich na sylaby. Łamaniem tekstu nie jest dzielenie słów w dowolnych miejscach.

5 Punktacja

Zadania 1-7 oceniane są na 1 punkt. Ostatnie zadanie jest za 3 punkty.