Lexikální analýza zdrojového kódu jazyka C

(195.113.207.171:32)

Cíl:

Osvojit si základní techniky lexikálního zpracování standardních zdrojových textů v jazyce C včetně elementární interakce s tabulkou symbolů.

Příklad 1: Tokenizace zdrojového kódu

Pomocí generátoru lexikálních analyzátorů *(f)lex* vytvořte program, rozpoznávající následující klíčová slova, operátory a identifikátory (souhrnně lexémy) jazyka C:

`main, int, void, return`
`{ } ( ) ; = + *`
`id` (proměnná), `intconst` (celočíselná konstanta)

Pro každý lexém definujte jeho obecnou výčtovou kategorii (token) a po nalezení uložte i jemu odpovídající atribut (value, pattern).
Jedná-li se např. o proměnnou, pak atributem je její jméno, u číselné konstanty zase odpovídající numerická hodnota. Váš lexikální analyzátor musí po zpracování kompletního vstupního textu (kódu) vypsat kategorii, odpovídající nalezenému lexému (*TOKEN*), *oddělovač*, atribut (*ATTRI*), *oddělovač* a odpovídající číslo řádku (*LINENO*). Nezapomeňte také řádně ošetřit kumulativní chybový stav, tj. vše, co neodpovídá výše uvedené specifikaci . Očekávaný výstup pro následující C kód:

```
int main(void)    {
      int x;
      x = 19;
      x = x * x;
      return      }
```

pak bude vypadat následovně:

```
      TOKEN           ATTRI   LINENO
      tok_int                 1
      tok_main                1
      tok_lparen              1
      tok_void                1
      tok_rparen              1
      tok_lbrace              2
      tok_int                 3
      tok_id          x       3
      tok_semicolon           3
      tok_id          x       4
      tok_equal               4
      tok_intconst    19      4
      tok_semicolon           4
      tok_id          x       5
      tok_equal               5
      tok_id          x       5
      tok_mult                5
      tok_id          x       5
      tok_semicolon           5
      tok_rbrace              7
      tok_return              6
      tok_intconst    0       6
      tok_semicolon           6
      tok_rbrace              7
      7 lines tokenized without problems. End of input reached.
```

Příklad 2: Instalace řetězců do tabulky symbolů

Navrhněte lexikální analyzátor pro zpracování nepojmenovaných řetězcových konstant (řetězcových literálů) v jazyce C, který uloží jejich nalezené hodnoty do jednoduché tabulky symbolů, realizované např. jako dvourozměrné pole. Po zpracování celého vstupního textu program vytiskne obsah tabulky symbolů. Předpokládejte, že každý nalezený literál musí být umístěn na zvláštním řádku.

V tomto příkladu se nebudeme zabývat speciálními formátovacími znaky (%s, %d apod.) a budeme je považovat za integrální součást hledaného řetězce. Jejich úplné zpracování by totiž dále komplikovalo a brzdilo lexikální analýzu. V praxi je proto obvyklé přesunout podobné relativně specifické a řídce se vyskytující činnosti až do fáze syntaktické analýzy (parsing). Při ukládání jednotlivých řetězců do tabulky symbolů (znaků) nezapomeňte v souladu se syntaxí jazyka C ukončit každý z nich znakem '\0' (NULL).

Většinu řetězců lze detekovat pomocí regulárního výrazu typu \".*\". Vy ale navíc:

- Korektně eliminujte nespárované uvozovky v rámci jednoho řádku,
- Umožněte vložení předznamenaného znaku uvozovky (\“) dovnitř literálu,

tak, aby po zpracování následujícího kódového fragmentu:

```
printf("Quotation mark: \" and other string: "); printf("Other string");
```

Váš program vypsal následující dva řádky (jeden vstup do tabulky symbolů pro každý nalezený řetězec):

```
Quotation mark: \" and other string:
Other string
```

Nepovinná část, dotovaná zvláštními body:

Pokud se Vám zdá navržené řešení poněkud těžkopádné, vylepšete ho následujícím způsobem:

- Uvozovky uvnitř literálů se budou zobrazovat bez prefixového zpětného lomítka,
- Zpětné lomítko využijete k rozdělení literálu na více bezprostředně následujících řádků. Zároveň ošetřete případ, kdy se uvnitř literálu vyskytne nepředznamenaný znak konce řádku,

tak, aby po zpracování následujícího kódového fragmentu:

```
printf("Quotation mark: \" and other \
string: "); printf("Other string");
```

Váš program vypsal následující dva řádky:

```
Quotation mark: " and other string:
Other string
```

K urychlení práce můžete využít rozpracovaná řešení *h5_1.l* a *h5_2.l* z disku S a odpovídající šablonu sestavovacího souboru *mf_X*. Chování jednotlivých úloh můžete následně otestovat na datových souborech *test.c, test1.txt a text2.txt*, které přesměrujete na standardní vstup lexikálního analyzátoru.