Добавил(а) Гісь
22.04.11 14:12 -

Синтаксичний розбір і лексичний аналіз виразів

Однієї з головних причин, що лежать в основі появи мов програмування високого рівня, з'явилися обчислювальні задачі, що вимагають великих об'ємів рутинних обчислень. Тому до мов програмування пред'являлися вимоги максимального наближення форми запису обчислень до природної мови математики. В зв'язка з цей одній з перших областей системного програмування сформувалося дослідження способів виразів. Тут отримані численні результати, проте найбільше розповсюдження отримав метод трансляції за допомогою зворотного польського запису, який запропонував польський математик Я. Лукашевіч.

Зворотна польська нотація

Нехай заданий простий арифметичний вираз вигляду:

(A+B)*(C+D)-E (1)

Представимо цей вираз у вигляді дерева, в якому вузлам відповідають операції, а гілкам - операнди. Побудову почнемо з кореня, як який вибирається операція, що виконується останньої. Лівій гілці відповідає лівий операнд операції, а правої гілки - правий. Дерево виразу (1) показано на рис. 1.

\- / / * E
/
/
/
/

+

+

/

/

/

/

A

B

C

D

рис. 1

Зробимо обхід дерева, під яким розумітимемо формування рядка символів з символів вузлів і гілок дерева. Обхід скоюватимемо від найлівішої гілки управо і вузол переписувати у вихідний рядок тільки після розгляду всіх його гілок. Результат обходу дерева має вигляд:

AB+CD+*E- (2)

Характерні особливості виразу (2) полягають в проходженні символів операцій за символами операндів і у відсутності дужок. Такий запис називається зворотним польським записом.

Зворотний польський запис володіє поряд чудових властивостей, які перетворюють її на ідеальну проміжну мову при трансляції. По-перше, обчислення виразу, записаного в зворотному польському записі, може проводитися шляхом однократного перегляду, що є вельми зручним при генерації об'єктного коду програм. наприклад, обчислення виразу (2) може бути проведено таким чином:

|-----|----------|----------| | # | Аналізований | Дія | | п/п |

рядок

|

|

|-----|----------|----------|

|

0

|

A B + C D + * E -

|

r1=A+B

|

|

1

|

r1
З
D + * E -

|

r2=C+D

|

|

2

|

r1 r2 * E -

|

r1=r1*r2

|

|

3

|

r1 E -

|

r1=r1-E

|

|

4

|

r1

|

Обчислення закінчено |

|-----|----------|----------|

Тут r1, r2 - допоміжні змінні.

По-друге, отримання зворотного польського запису з початкового виразу може

здійснюватися вельми просто на основі простого алгоритму, запропонованого Дейкстрой. Для цього вводиться поняття стекового пріоритету операцій(табл. 1):

Таблиця 1 |----------|----------| | Операція | Пріоритет | |----------|----------| | (| 0

|

|

)

|

1

|

|

+|-

|

2

|

|

*|/

|

3

|

|

**

|

4

|

|----------|----------|

Є видимим початковий рядок символів зліва направо, операнди переписуються у вихідний рядок, а знаки операцій заносяться в стек на основі наступних міркувань:

а) якщо стек порожній, то операція з вхідного рядка переписується в стек;

б) операція виштовхує із стека всі операції з великим або рівним пріоритетом у вихідний рядок;

в) якщо черговий символ з початкового рядка є відкриваюча дужка, то він проштовхується в стек;

г) закриваюча кругла дужка виштовхує всі операції із стека до найближчої відкриваючої дужки, самі дужки у вихідний рядок не переписуються, а знищують один одного.

Процес отримання зворотного польського запису виразу (1) схемно представлений :

Символ
1
2
3

4

5

6

7

8

9

10

11

12

13

Вхідний рядок

(

A

+

B

)

*

(

C

+

D

)

-

E

Стан стека

(

(

+
(

+
(

*

(

*

(
*

+
(
*

+
(
*

*

-

-

Вихідний
рядок

A

B

+

C

D

+

*

E

-

Розбір арифметичного(і не тільки) виразу. Класичні алгоритми.

Алгоритм Рутісхаузера

Даний алгоритм є одним з найстаріших. Його особливістю є припущення про повну дужкову структуру виразу. Під повним дужковим записом виразу розуміється запис, в якому порядок дій задається розстановкою дужок. Неявне старшинство операцій при цьому не враховується. Наприклад:

D:=((C-(B*L))+K)

Обробляючи вираз з повною дужковою структурою, алгоритм привласнює кожному символу з рядка номер рівня за наступним правилом:

1. якщо це дужка або змінна, що відкривається, то значення збільшується на 1;

2. якщо знак операції або дужка, що закривається, то зменшується на 1.

Для виразу (A+(B+C)) привласнення значень рівня відбуватиметься таким чином :

|-------|---------------------------------------| |N симв.| 1 2 3 4 5 6 7 8 9 | |-------|---------------------------------------|

|Символи|

|

|рядка

| (

A

+

(

B

*

З

)

)

|

|-------|---------------------------------------|

|Номера |

|

|рівнів | 1

2

1

2

3

2

3

2

1

|

|-------|--------------------------------------|

Алгоритм складається з наступних кроків:

1. виконати розстановку рівнів;

2. виконати пошук елементів рядка з максимальним значенням рівня;

3. виділити трійку - два операнди з максимальним значенням рівня і операцію, яка укладена між ними;

4. результат обчислення трійки позначити допоміжній змінній;

5. з початкового рядка видалити виділену трійку разом з її дужками, а на її місце помістити допоміжну змінну, що позначає результат, із значенням рівня на одиницю менше ніж у виділеної трійки;

6. виконувати п.п.2 - 5 до тих пір, поки у вхідному рядку не залишиться одна змінна, що позначає загальний результат виразу.

Приклад розбору :

|----------|----------------------------------| |Генеруючі | Вираз
|
|
т
рійки

|

|

|----------|----------------------------------|

|

|

((((A + B) * З) / D) - E)

|

|

| 0 1 2 3 4 5 4 5 4 3 4 3 2 3 2 1 2 1 0|

|

|

|

|+ A B - > R |

(((R * З) / D) - E)

|

|

|

0 1 2 3 4 3 4 3 2 3 2 1 2 1 0

|

|

|

|

|* R
З
-> S

|

((S / D) - E)

|

|

|

0 1 2 3 2 3 2 1 2 1 0

|

|

|

|

|------------|-------------------------------------|

|------------|-------------------------------------|

|Генерируючі |

Вираз

|

|

трійки

|

|

|------------|-------------------------------------|

|/ S D -> Q

|

(Q - E)

|

|

|

0 1 2 1 2 1 0

|

|

|

|

|- Q E -> T

|

T

|

|----------|----------------------------------|

Алгоритм Бауера і Замельзона

З ранніх стекових методів розглядається алгоритм Бауера і Замельзона. Алгоритм використовує два стеки і таблицю функцій переходу. Один стек використовується при трансляції виразу, а другий - під час інтерпретації виразу. Позначення: T - стек транслятора, E - стек інтерпретатора.

В таблиці переходів задаються функції, які повинен виконати транслятор при розборі виразу. Можливі шість функцій при прочитанні операції з вхідного рядка:

- f1 - заслати операцію з вхідного рядка в стек T; читати наступний символ рядка;

- f2 - виділити трійку - узяти операцію з вершини стека T і два операнди з вершини стека E; допоміжну змінну, що позначає результат, занести в стек E; заслати операцію з вхідного рядка в стек T; читати наступний символ рядка;

- f3 - виключити символ із стека T; читати наступний символ рядка;

- f4 - виділити трійку - узяти операцію з вершини стека T і два операнди з вершини стека E; допоміжну змінну, що позначає результат, занести в стек E; по таблиці визначити функцію для даного символу вхідного рядка;

- f5 - видача повідомлення про помилку;

- f6 - завершення роботи.

Таблиця переходів для виразів алгебри матиме вигляд(символ $ є ознакою порожнього стека або порожнього рядка):

|----------|----------| | Операція з вхідного рядка | | |----------|

|

|

$

(

+

-

*

/

)

|

|----------|---|----------|

|Операція

|$

| 6

1

1

1

1

1

5

|

|на вершині|(

| 5

1

1

1

1

1

3

|

|стека T

|+

| 4

1

2

2

1

1

4

|

|

|-

| 4

1

2

2

1

1

4

|

|

|*

| 4

1

4

4

2

2

4

|

|

|/

| 4

1

4

4

2

2

4

|

|----------|---|----------------------------|

Алгоритм проглядає зліва направо вираз і циклічно виконує наступні дії: якщо черговий символ вхідного рядка є операндом, то він безумовно переноситься в стек E; якщо ж операція, то по таблиці функцій переходу визначається номер функції для виконання. Для виразу A+(B-C)*D нижче приводиться послідовність дій алгоритму.

|-------|--------|-------|-------|----------| |стек E | стік T |Вхідний|Номер | Трійка | | |

|символ |функції|

|

|-------|--------|-------|-------|----------|

|$

| $

|

A

|

|

|

|$A

| $

|

+

|

1

|

|

|$A

| $+

|

(

|

1

|

|

|$A

| $+(

|

B

|

|

|

|$AB

| $+(

|

-

|

1

|

|

|$AB

| $+(-

|

З

|

|

|

|$ABC

| $+(-

|

)

|

4

|- B
3
->R |

|$AR

| $+(

|

)

|

3

|

|

|$AR

| $+

|

*

|

1

|

|

|$AR

| $+*

|

D

|

|

|

|$ARD

| $+*

|

$

|

4

|* R D ->Q |

|$AQ

| $+

|

$

|

4

|+
A
Q ->S |

|$S

| $

|

$

|Кінец

|

|

|-------|--------|-------|-------|----------|

Задачі

Задача GoTo.

Учні, недавно що почали програмувати, вживають дуже багато операторів GOTO, що є майже неприпустимим для структурованої програми. Допоможіть викладачу інформатики написати програму, яка оцінюватиме ступінь структурованості відладженої програми школяра на мові Pascal, спершу просто підраховувавши кількість операторів GOTO в ній.

В синтаксично вірній програмі ключове слово оператора переходу GOTO може стояти або на початку рядка або після пропуску або після одного з символів — ‘;’, ‘:’, ‘}’, а після нього може стояти або пропуск або переклад рядка або символ ‘{‘ (табуляцію як

роздільник розглядати не будемо).

Нагадаємо, що окрім позначення дійсно оператора переходу, слово GOTO може зустрічатися в тексті програми в рядкових константах, укладених в апострофи, або в коментарях, причому для простоти вважатимемо, що коментар завжди починається з символу ‘{‘, а закінчується першим що зустрівся після цього символом ‘}’, при цьому символ ‘{‘ повинен знаходитися не усередині рядкової константи. В цих випадках слово GOTO підраховувати не потрібно. Рядкові і прописні букви в Pascal не помітні.

У вхідному файлі goto.in знаходиться текст програми без синтаксичних помилок на мові Pascal. Розмір програми не перевершує 64K. У вихідному файлі goto.out повинне виявитися одне число – кількість операторів GOTO в цій програмі.

Приклад вхідного файлу:

```
label 1,2;

var I:byte;

begin

 1: I:=I+1;

    if I>10 then goto 2 else

      writeln(¢ goto ¢); GoTo 1;
```

2: if I