Пример отчета по лабораторным работам по автоматизированной разработке синтаксических анализаторов
Синтаксический анализ скобочных выражений

Текст задания: разработать программу синтаксического анализатора (далее по тексту - программа BRACKET) для проверки расстановки скобок в математических выражениях.

Введение

Настоящий документ определяет техническое задание на разработку программы синтаксического анализатора (далее по тексту - программа BRACKET) для проверки расстановки скобок в математических выражениях.

Основания для разработки

Программа BRACKET разрабатывается в рамках лабораторной работы по курсу "*Программное обеспечение САПР*" для практического изучения этапа синтаксического анализа в процедурах трансляции формальных языков.

Назначение разработки

Программа BRACKET предназначается для реализации грамматического разбора скобочных конструкций в пакетах символической обработки математических формул.

Требования к программе

1. Требования к функциональным характеристикам

1.1. Программа BRACKET должна обеспечивать синтаксический анализ расстановки скобок в произвольном математическом выражении, которое передается символьной строкой из потока стандартного ввода.

1.2. Программа BRACKET должна предусматривать возможность обработки 3-х типов открывающих и закрывающих скобок: фигурные, квадратные и круглые, которые могут одновременно присутствовать и повторяться во входном выражении в прозвольном числе экземпляров.

1.3. Программа BRACKET должна осуществлять количественный контроль соответствия числа открывающих и закрывающих скобок каждого типа.

1.4. Программа BRACKET должна проверять, что каждой закрывающей скобке любого типа предшествует открывающая скобка того же типа.

1.5. Программа BRACKET должна учитывать общепринятые математические правила взаимного расположения скобок различных типов, в частности, круглые скобки не могут окружать квадратные или фигурные, а внутри любой пары квадратных скобок не должно быть фигурных.

1.6. Программа BRACKET должна обнаруживать любые нарушения порядка расположения скобок, которые противоречат требованиям пп. 1.3 - 1.5.

1.7. Любые нарушения допустимого порядка расположения скобок во входной строке из потока стандартного ввода должны сопровождаться информационным сообщением "*** syntax error" в потоке стандартного вывода. Диагностика ошибки должна содержать порядковый номер первой скобки, расположение которой противоречит требованиям пп. 1.3 - 1.5.

1.8. Результат распознавания программой BRACKET корректного скобочного выражения должен сопровождаться отображением в потоке стандартного вывода информационным сообщением "Correct Bracket Expression".

2. Требования к надежности

2.1. Программа BRACKET не должна иметь каких-либо ограничений по числу скобок и других символов в анализируемом выражении, кроме внутренних ограничений инструментальных средств, использованных для ее реализации.

2.2. Программа BRACKET не должна иметь каких-либо ограничений по уровню вложенности и числу последовательных скобочных блоков, кроме внутренних ограничений инструментальных средств, использованных для ее реализации.

3. Требования к составу и параметрам технических средств

Программа BRACKET должна быть разработана исходя из возможности реализации на стандартном составе технических средств компьютеров любой архитектуры, после соответствующей трансляции исходного кода.

4. Условия эксплуатации

4.1. Программа BRACKET должна быть ориентирована на эксплуатацию в операционной среде OS UNIX любой версии.

4.2. Программа BRACKET должна быть реализована в виде выполняемого файла с именем bracket, по которому она должна вызываться средствами любого командного процессора OS UNIX.

4.3. Программа BRACKET должна эксплуатироваться в интерактивном режиме, читая строку входного выражения из потока стандартного ввода и отображая результат ее обработки в потоке стандартного вывода.

5. Требования к информационной и программной совместимости

5.1. Чтобы обеспечить выполнение требуемых технических характеристик, программа BRACKET должна реализовывать синтаксический анализ любой входной строки скобочного выражения из потока стандартного ввода.

5.2. Синтаксический анализатор программы BRACKET должен обеспечивать грамматический разбор скобок с целью установить соответствие или несоответствие содержащей их строки потока стандартного ввода требуемому формату скобочного выражения.

5.3. Для выполнения грамматического разбора синтаксический анализатор программы BRACKET должен реализовывать однозначную контекстно-свободную грамматику простого предшествования, которая ориентирована на обработку строки заданного скобочного выражения из потока стандартного ввода и далее по тексту называется грамматикой скобок.

5.4. Грамматику скобок синтаксического анализатора программы BRACKET должны составлять следующие элементы:

- терминальные символы (терминалы), соответствующие структурным единицам (лексемам) входного скобочного выражения;
- начальный нетерминальный символ (начальный нетерминал), к которому приводится входное скобочное выражение;
- нетерминальные символы (нетерминалы), обозначающие допустимые варианты комбинации лексем во входном скобочном выражении;
- система продукций (правил вывода), обеспечивающая грамматический разбор входного скобочного выражения.

5.5. Терминальные символы грамматики скобок синтаксического анализатора программы BRACKET должны представляться лексемами, специфицированными следующими литералами:

'(', ')', '[', ']', '{', '}',

которые обозначают коды ASCII соответствующих круглых, квадратных и фигурных открывающих или закрывающих скобок.

5.6. Нетерминальные символы грамматики скобок должны обозначаться следующими именами:

p, pp, s, sp, spps, b, x, xx.

Они должны определяться продукциями грамматики скобок синтаксического анализатора программы BRACKET в форме, близкой нотации Бэкуса-Наура (БНФ), следующим образом:

```
 p : блок круглых (Parentheses) скобок;
pp :  последовательность блоков круглых скобок;
 s : блок квадратных (Square) скобок;
sp : блок квадратных или круглых скобок;
spps : последовательность блоков круглых и/или квадратных скобок;
 b : блок фигурных (Braces) скобок;
 x : блок любых (круглых, квадратных или фигурных) скобок;
xx : последовательность произвольных скобочных блоков
```

5.7. Начальный нетерминал грамматики скобок синтаксического анализатора программы BRACKET должен обозначаться именем inputline. Он должен выводиться из любой корректной последовательности произвольных скобочных блоков. Грамматический вывод начального нетерминала из любой входной строки, содержащей корректную расстановку скобок, должна обеспечивать система продукций грамматики скобок.

5.8. Система продукций (правил вывода) грамматики скобок синтаксического анализатора программы BRACKET должна обеспечивать грамматический разбор произвольной входной строки потока стандартного ввода путем приведения терминалов и нетерминалов к начальному нетерминалу методом LR(1) анализа снизу-вверх по дереву вывода. Приведение входной строки к начальному нетерминалу по правилам вывода грамматики скобок должно означать успех грамматического разбора скобочного выражения, т.е. распознавание корректного скобочного выражения. Альтернативный результат должен рассматриваться как синтаксическая ошибка расстановки скобок во введенной строке.

5.9. Для разработки синтаксического анализатора программы BRACKET, необходимо использовать генератор синтаксических анализаторов (далее по тексту - YACC) из состава OS UNIX, инструментальные средства которого ориентированы на обработку файла спецификаций (далее по тексту, Yacc-файл) проектируемого синтаксического анализатора.

5.10. При разработке синтаксического анализатора программы BRACKET необходимо составить Yacc-файл, отражающий специфику грамматического разбора скобочного выражения, и сохранить его под именем bracket.y в выбранном доступном рабочем каталоге файловой системы OS UNIX.

5.11. Проектируемый Yacc-файл bracket.y должен состоять из 3-х секций: деклараций, правил и функций. Разделителем секций должны быть символические пары %%, расположенные в начальных позициях содержащих их строк Yacc-файла. Каждая секция Yacc-файла bracket.y должна содержать соответствующее число комментариев в формате, принятом для системы программирования C.

5.12. Секция деклараций Yacc-файла bracket.y должна включать:

- объявление начального нетерминала грамматики скобок с помощью директивы %Start;
- спецификацию блока внешних описаний, ограниченную директивами %{ и %}, в котором необходимо декларировать и определить внешнюю (extern) целочисленную (типа int) переменную yylval для фиксации порядковых номеров скобок в скобочном выражении.

5.13. В секции правил Yacc-файла bracket.y должны быть приведены описания продукций приведения нетерминалов грамматики скобок в соответсвии с требованием пп. 5.5 - 5.8.

5.14. Каждая продукция секции правил Yacc-файла bracket.y должна быть задана в нотации, близкой к форме Бэкуса-Наура, где в левой части указывается приводимый нетерминал, а в правой - последовательность терминалов и/или нетерминалов грамматики скобок, которые перечисляются через пробел. Для разделения частей продукции должен использоваться символ двоеточия (:). Каждую продукцию нужно начинать с новой строки и завершать либо символом точки с запятой (;), либо блоком действий в фигурных скобках. Например, начальный нетерминал inputline может определяться следующим правилом:

```
inputline : xx ;
```

которое позволяет трактовать inputline как последовательность произвольных скобочных блоков, свернутую в нетерминал xx.

5.15. Продукции секции правил Yacc-файла bracket.y, приведение нетерминалов которых необходимо сопровождать функциональной обработкой, должны содержать блоки действий. Блоки действий должны располагаться в правых частях продукций и ограничиваться парой фигурных скобок. Внутри блоков действий можно использовать любые конструкции и вызовы функций системы программирования C. Например, правило определения начального нетерминала inputline (см. п. 5.14) целесообразно дополнить следующим блоком действий:

```
inputline : xx { /* Grammar Success */
                 puts("Correct Bracket Expression");
                 return(0);
               }
```

который обеспечивает требуемую функциональную обработку при успехе грамматического разбора входной строки (см. п. 1.8).

5.16. Продукции секции правил Yacc-файла bracket.y, необходимые для приведения нетерминалов: p, s, b определяющих одиночные блоки скобок соответствующих типов (см. п. 5.6), должны быть специфицированы с помощью альтернативных правил. В каждом из них альтернативы свертки различных нетерминалов правой части должны быть об'единены с помощью оператора ИЛИ, который обозначается символом вертикальной черты (|). Например, альтернативную продукцию приведения нетерминала b следует реализовать путем сочетания 2-х правил:

```
b : '{' '}' ;      /* rule b.1 */
b : '{' xx '}' ;   /* rule b.2 */
```

в форме одного альтернативного правила:

```
/* Braces block Alt Rule */
b : '{'  '}'   |     /* alt-rule b.1 */
    '{' xx '}' ;     /* alt-rule b.2 */
```

которое определяет нетерминал b как блок фигурных скобок, содержащий вложенную последовательность блоков любых скобок, либо как пустой блок фигурных скобок без каких-либо скобочных символов внутри. Очевидно, что блоки круглых (p) и квадратных (s) могут определяться альтернативными правилами аналогичной структуры.

5.17. Продукции секции правил Yacc-файла bracket.y, необходимые для приведения нетерминалов: pp, spps, xx, определяющих последовательности блоков скобок соответствующих типов (см. п. 5.6), должны задаваться в форме лево-рекурсивных правил. Например, нетерминал xx рекомендуется специфицировать следующим лево-рекурсивным правилом:

```
/* Arbitrary brackets' blocks sequence */
xx : x    |
     xx x ;
```

которое позволяет трактовать нетерминал xx как одиночный блок любых скобок (x), или (|) последовательность таких блоков произвольной длины. Аналогичные лево-рекурсивные правила следует составить для приведения последовательности блоков круглых скобок (pp) и комбинированной последовательности из блоков круглых и/или квадратных скобок (spps).

5.18. Чтобы иметь возможность абстрагироваться от типа скобок, образующих одиночный скобочный блок, в лево-рекурсивных продукциях секции правил Yacc-файла bracket.y (см. п. 5.17) необходимо предусмотреть альтернативные правила разименования, которые должны обеспечивать приведение нетерминалов sp и x (см. п. 5.6). Например, нетерминал x целесообразно определить следующим альтернативным правилом:

```
/* Arbitrary brackets block */
x : p | s | b ;
```

согласно которому, к нетерминалу x приводится любой скобочный блок, вне независимости от типа образующих его скобок. Аналогичное по структуре альтернативное правило разименования следует предусмотреть для приведения нетерминала sp, определяющее его как блок квадратных (s) или круглых (p) скобок.

5.19. Секция функций Yacc-файла bracket.y должна содержать исходный код, оформленный по правилам системы программирования C, для 3-х функций с предопределенными именами: yylex(), yyerror() и main(), которые должны иметь целочисленный (типа int) код возврата. Исходный код перечисленных функций используется непосредственно для формирования кода программы BRACKET.

5.20. Функция yylex() секции функций Yacc-файла bracket.y должна выполнять лексический анализ входной строки из потока стандартного ввода со следующими целями: обнаружить скобки, распознать типы скобок и зафиксировать порядковые номера скобок. Вызов функции yylex() будет автоматически производиться при грамматическом разборе входной строки для получения очередных скобок, пока не достигнут символ перевода строки ('\n').

5.21. Для достижения целей лексического анализа, указанных в п. 5.20, исходный код функции yylex() должен предусматривать:

- побайтное чтение любой заданной строки входного скобочного выражения из потока стандартного ввода, например, с помощью библиотечной функции getchar() системы программирования C;
- распознавание символов скобок во входной строке например, с помощью оператора switch системы программирования C;

- возврат кодов ASCII обнаруженных скобок с помощью оператора return системы программирования C;
- возврат нулевого кода при достижении конца входной строки с помощью оператора return(0) системы программирования C;
- использование значения внешней целочисленной переменной yylval (см. п. 5.12) для фиксации порядкового номера каждой очередной обнаруженной скобки.

5.22. Функция yyerror() секции функций Yacc-файла bracket.y должна обеспечивать обработку ошибок расстановки скобок, обнаруженных при грамматическом разборе входной строки скобочного выражения. Ее вызов будет происходить автоматически, когда скобочное выражение не может быть приведено к начальному нетерминальному символу грамматики скобок, обеспечивая аварийное прерывание грамматического разбора текущей входной строки потока стандартного ввода.

5.23. Спецификация функции yyerror() в Yacc-файле bracket.y, должна иметь единственный аргумент типа (char *), который по умолчанию содержит адрес предопределенной символьной строки: "syntax error". Аргумент функции yyerror() следует использовать для формирования диагностического сообщения в соответствии с требованием п. 1.7. Для диагностики позиции ошибки по номеру неверно установленной скобки, исходный код функции yyerror() должен предусматривать использование значения внешней целочисленной переменной yylval. Для отображения диагностического сообщения в потоке стандартного вывода, рекомендуется применить библиотечную функцию printf() системы программирования C.

5.24. Спецификация функции main() в секции функций Yacc-файла bracket.y должна содержать исходный код основной функции программы BRACKET, который обеспечивает обращение к функции синтаксического анализа yyparse() и передачу ее кода возврата во внешнюю операционную среду. Исходный код функции yyparse() формируется генератором синтаксических анализаторов YACC по Yacc-файлу и предусматривает вызов функции yylex() для выполнения лексического анализа, также как передачу управления функции yyerror() для обработки ошибок грамматического разбора. Код возврата функции yyparse() должен однозначно отражать результат грамматического разбора, принимая значение 0 при его успехе, или 1 - при обнаружении синтаксических ошибок.

5.25. Исходный код секции функций Yacc-файла bracket.y и функции yyparse() образует исходный код программы BRACKET, который должен формироваться генератором синтаксических анализаторов YACC в файле с предопределенным именем y.tab.c в текущем рабочем каталоге файловой системы OS UNIX. Выполняемый модуль программы BRACKET должен быть создан по файлу исходного кода y.tab.c в выполняемом файле bracket средствами компилирующей системы программирования C.

Стадии и этапы разработки

Процесс разработки программы BRACKET целесообразно разделяться на следующие 3 стадии:

- составить Yacc-файл bracket.y в выбранном рабочем каталоге файловой системы OS UNIX, используя любой текстовый редактор, например, xedit, ориентированный на работу в операционной среде X Window System, или joe, для редактирования в консольном режиме;
- получить исходный код синтаксического анализатора в файле y.tab.c текущего каталога файловой системы OS UNIX, обработав Yacc-файл bracket.y командой yacc, следующим образом:

```
$ yacc bracket.y
```

- сформировать выполняемый модуль в файле bracket текущего каталога файловой системы OS UNIX, компилируя исходный код синтаксического анализатора в файле y.tab.c следующей командой:

```
$ cc -o bracket y.tab.c
```

Результаты разработки программы BRACKET необходимо оформить в виде проектного документа, содержащего описание грамматики скобок и файла спецификаций для генератора синтаксических анализаторов YACC.

Порядок контроля и приемки

1. Для проверки функционирования программы BRACKET должны быть предложены контрольные примеры, предусматривающие стандартный ввод корректных и некорректных скобочных выражений с учетом вложенности и конкатенации блоков скобок различных типов.
2. Для приемки программы BRACKET должен быть организован вызов выполняемого файла bracket в консольном режиме работы OS UNIX или режиме эмуляции терминала операционной среды X Window System.

Приложение1

Текст высокоуровневой спецификации.

Приложение2

При разработке синтаксических анализаторов в OS UNIX для проектирования программы BRACKET рекомендуется использовать литературные источники, перечисленные ниже.

1. Рейуорд-Смит В.Дж. Теория формальных языков. Вводный курс, М.: Радио и связь, 1988.
2. Тихомиров В.П., Давидов М.И. Операционная система ДЕМОС: инструментальные средства программирования, М.: Финансы и статистика, 1988.

3. SCO XENIX, Development System, YACC Programmer Guide, SCO Inc., 1986. (Имеется русский перевод: Генератор синтаксических анализаторов YACC. Руководство Программиста)