ГУАП

КАФЕДРА № 43

ОТЧЕТ   
ЗАЩИЩЕН С ОЦЕНКОЙ

ПРЕПОДАВАТЕЛЬ

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| профессор, к. т. н., профессор |  |  |  | М.И. Гвоздик |
| должность, уч. степень, звание |  | подпись, дата |  | инициалы, фамилия |

|  |
| --- |
| ОТЧЕТ О ЛАБОРАТОРНОЙ РАБОТЕ №2 |
| ВОСХОДЯЩАЯ ТРАНСЛЯЦИЯ НА ОСНОВЕ ВЫЧИСЛЕНИЯ СИНТЕЗИРУЕМЫХ АТРИБУТОВ |
| по курсу: МЕТОДЫ ТРАНСЛЯЦИИ |
|  |
|  |

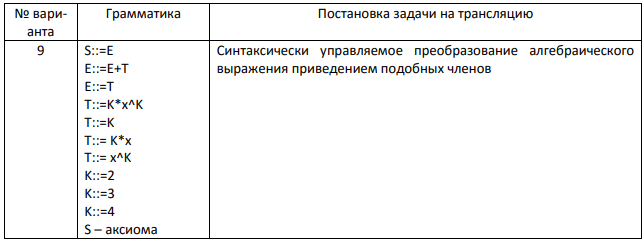
РАБОТУ ВЫПОЛНИЛ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| СТУДЕНТ ГР. № | 4631 |  |  |  | К.С. Ларионов |
|  |  |  | подпись, дата |  | инициалы, фамилия |

Санкт-Петербург 2018

**Задание**

Дополните программу, разработанную в лабораторной работе №1, вычислением синтезируемых атрибутов для программной реализации синтаксически управляемого решения задачи, выбранной по номеру варианта.

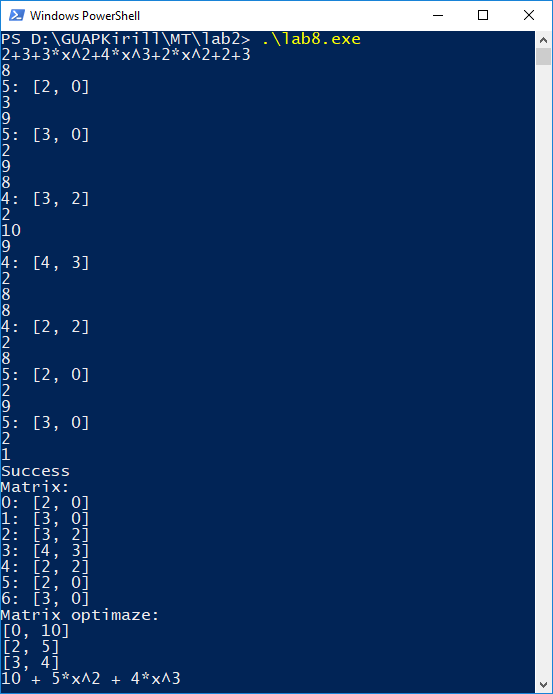


Определим функции вычисления синтезируемого атрибута для левой части правила, имея в виду, что значением этого атрибута должно быть натуральное число, равное показателю степени или коэффициенту перед *x* найденных к моменту применения соответствующих правил. Данные при разборе записываются в матрицу для последующего преобразования алгебраического выражения приведением побочных членов.

|  |
| --- |
|  |

|  |
| --- |
| Lab2.l |
| %option noyywrap  %{  #include <stdio.h>  #define YY\_DECL int yylex()  #include "y.tab.h"  %}  %%  \n {return T\_NEWLINE;}  2 {return '2';}  3 {return '3';}  4 {return '4';}  "\*" {return '\*';}  "+" {return '+';}  "^" {return '^';}  "x" {return 'x';}  %% |
| Lab2.y |
| %{  #include <stdio.h>  #include <string.h>  #include <stdlib.h>  extern int yylex();  extern int yyparse();  extern FILE\* yyin;  void yyerror(const char\* s);  int mtr[10][2];  int mtrNext[5];  int idx = 0;  void RES(){    int i;  int j;  printf("Success\n");  printf("Matrix:\n");  for(i = 0; i< 10;i++) {  if (!(mtr[i][0] == 0 && mtr[i][1] == 0)) {  printf("%d: [%d, %d]\n", i, mtr[i][0], mtr[i][1]);  }  }  for(i = 0; i< 10;i++) {  switch(mtr[i][1])  {  case 0: {  mtrNext[0] += mtr[i][0];  break;  }  case 1: {  mtrNext[1] += mtr[i][0];  break;  }  case 2: {  mtrNext[2] += mtr[i][0];  break;  }  case 3: {  mtrNext[3] += mtr[i][0];  break;  }  case 4: {  mtrNext[4] += mtr[i][0];  break;  }  }  }  printf("Matrix optimaze:\n");  for(i = 0; i< 5;i++) {  if(mtrNext[i] != 0)  printf("[%d, %d]\n", i, mtrNext[i], i);  }  j = 0;  for(i = 0; i< 5;i++) {  if(mtrNext[i] == 0) continue;  if (j != 0) printf(" + ");    if( i == 0 )  printf("%d", mtrNext[i]);  else if( i == 1 )  printf("%d\*x", mtrNext[i]);  else  printf("%d\*x^%d", mtrNext[i], i);  j++;  }  idx = 0;  for(i = 0; i< 5; i++)  mtrNext[i] = 0;      for(i = 0; i< 10; i++) {  mtr[i][0] = 0;  mtr[i][1] = 0;  }  };  %}  %token T\_NEWLINE  %start calculation  %%  calculation: | calculation line;  line: T\_NEWLINE  | S T\_NEWLINE { RES(); } ;  S: E { printf("1\n"); };  E: E '+' T { printf("2\n"); };  E: T { printf("3\n"); };  T: K '\*' 'x' '^' K { printf("4: [%d, %d]\n", $1, $5); mtr[idx][0] = $1; mtr[idx++][1] = $5; };  T: K { printf("5: [%d, 0]\n", $1); mtr[idx][0] = $1; mtr[idx++][1] = 0; };  T: K '\*' 'x' { printf("6: [%d, 1]\n", $1); mtr[idx][0] = $1; mtr[idx++][1] = 1; };  T: 'x' '^' K { printf("7: [1, %d]\n", $3); mtr[idx][0] = 1; mtr[idx++][1] = $3; };  K: '2' { $$ = 2; printf("8\n"); };  K: '3' { $$ = 3; printf("9\n"); };  K: '4' { $$ = 4; printf("10\n"); };  %%  int main() {  yyparse();  return 0;  }  void yyerror(const char\* s) {  fprintf(stderr, "Parse error: %s\n", s);  exit(1);  } |

**Пример работы:**



**Вывод**

Дополнил программу, разработанную в лабораторной работе №1, вычислением синтезируемых атрибутов для программной реализации синтаксически управляемого решения задачи.