**Compiler**

과제 9 : Using Bison with C++

이름 : 제정륭

학번 : 201224537

att.l

|  |
| --- |
| %{  #include "ast.h"  #include "y.tab.h"  #include <stdlib.h>  %}  %%  [0-9]+ { yylval.iVal = atoi(yytext); return NUM; }  [a-zA-Z]+ { yylval.sVal = yytext; return ID; }  [ \t] ;  \n return 0;  "+" return '+';  "-" return '-';  "\*" return '\*';  "/" return '/';  "(" return '(';  ")" return ')';  . { printf("error near %s\n", yytext); exit(-1); }  %%  int yywrap() {  return 1;  } |

* 수식에서 ‘x’와 같은 문자를 입력받기 위해 알파벳을 ID로 리턴하는 라인을 추가했습니다.

att.y

|  |
| --- |
| %{ #include <stdio.h>  #include <ctype.h>  #include "ast.h"  int yyerror(const char \*msg), yylex();  %}  %union {  Node \*pVal;  int iVal;  char \*sVal;  }  %type <pVal> Exp  %token <iVal> NUM  %token <sVal> ID  %left '-' '+'  %left '\*' '/'  %nonassoc UNARYMINUS  %%  Line : Exp { $1->printTree(0); }  ;  Exp : Exp '+' Exp { $$ = new Node(PLUS, 0, NULL, $1);  $1->bro = $3; }  | Exp '-' Exp { $$ = new Node(MINUS, 0, NULL, $1);  $1->bro = $3;}  | Exp '\*' Exp { $$ = new Node(MUL, 0, NULL, $1);  $1->bro = $3;}  | Exp '/' Exp { $$ = new Node(DIVIDE, 0, NULL, $1);  $1->bro = $3;}  | '-' Exp %prec UNARYMINUS { $$ = new Node(UMINUS, 0, NULL, $2); }  | '(' Exp ')' { $$ = $2; }  | NUM { $$ = new Node(INT, $1); }  | ID { $$ = new Node(CHAR, $1); }  ;  %%  int main() { yyparse(); return 0; }  int yyerror(const char \*msg) { fputs(msg, stderr); }  /\*  int yylex() { int c = getchar();  return (c == '\n')? 0: (isdigit(c))? yylval=c-'0', NUMBER: c;  }  \*/ |

* C++문법에 맞게 클래스 생성자를 사용해서 구현했습니다.

ast.h

|  |
| --- |
| #ifndef \_\_AST\_H\_\_  #define \_\_AST\_H\_\_  enum NKind{ PLUS, MINUS, MUL, DIVIDE, UMINUS, INT, CHAR };  class Node{  public:  Node(NKind kind, int val, Node \*bro, Node \*son);  Node(NKind kind, int val);  Node(NKind kind, char \*val);  NKind kind;  union{  int ival;  char \*sval;  };  Node \*bro, \*son;  void printTree(int depth);  };  #endif |

* Node클래스를 선언하고 세종류의 생성자와 멤버변수, 멤버함수를 넣었습니다.

ast.cpp

|  |
| --- |
| #include <stdio.h>  #include <stdlib.h>  #include <string.h>  #include <string>  #include <iostream>  #include <ctype.h>  #include "ast.h"  using namespace std;  void Node::printTree(int depth) {  int i;  if(this == NULL)  return;  switch(this->kind){  case PLUS :  for(i=0; i<depth; ++i)  printf("\t");  printf("Op(+)\n");  this->son->printTree(depth+1);  this->bro->printTree(depth);  break;  case MINUS :  for(i=0; i<depth; ++i)  printf("\t");  printf("Op(-)\n");  this->son->printTree(depth+1);  this->bro->printTree(depth);  break;  case MUL :  for(i=0; i<depth; ++i)  printf("\t");  printf("Op(\*)\n");  this->son->printTree(depth+1);  this->bro->printTree(depth);  break;  case DIVIDE :  for(i=0; i<depth; ++i)  printf("\t");  printf("Op(/)\n");  this->son->printTree(depth+1);  this->bro->printTree(depth);  break;  case UMINUS :  for(i=0; i<depth; ++i)  printf("\t");  printf("Uop(-)\n");  this->son->printTree(depth+1);  this->bro->printTree(depth);  break;  case INT :  for(i=0; i<depth; ++i)  printf("\t");  printf("Int(%d)\n",this->ival);  this->son->printTree(depth+1);  this->bro->printTree(depth);  break;  case CHAR :  for(i=0; i<depth; ++i)  printf("\t");  cout << "ID(" << \*(this->sval) << ")" << endl;  this->son->printTree(depth+1);  this->bro->printTree(depth);  break;  }  }  Node::Node(NKind kind, int val, Node \*bro, Node \*son){  this->kind = kind;  this->ival = val;  this->bro = bro;  this->son = son;  };  Node::Node(NKind kind, int val){  this->kind = kind;  this->ival = val;  this->bro = this->son = NULL;  };  Node::Node(NKind kind, char \*val){  this->kind = kind;  this->sval = val;  this->bro = this->son = NULL;  }; |

* 트리는 노드의 속성에 따라서 recursive하게 출력하도록 만들었습니다.
* Tree의 depth는 argument로 recursive하게 반복적으로 전달해서 해당 노드의 depth만큼 들여쓰기하도록 만들었습니다.
* Node의 생성자를 오버로딩해서 argument에 따라 다른 종류의 Node를 초기화 하도록 만들었습니다.