Project 2

Parser & Yacc

2013011509 오승준

1. 과제 개요

간단한 언어인 C-minus 언어를 기반으로 하는 parser를 Yacc를 이용해서 구현

**주어진 자료에 hint로 존재하는 부분은 보고서에 작성하지 않았습니다

2. 개발환경

1. Linux-16.04 LTS
2. Using GCC

3. 코드분석

- globals.h -

```
typedef enum {StmtK,ExpK,DecK,ParamK,TypeK} NodeKind;
typedef enum {CompK,IfK,IterK,RetK, ReadK, WriteK, RepeatK} StmtKind;
typedef enum {AssignK,OpK,ConstK,IdK,ArrIdK,CallK} ExpKind;
typedef enum {FuncK,VarK,ArrVarK} DecKind;
typedef enum {ArrParamK,NonArrParamK} ParamKind;
typedef enum {TypeNameK} TypeKind;
```

사진_1 주어진 규칙대로 사용될 state들의 이름

사진_1과 같이 사용되어질 변수들을 선언해준다.

```
typedef struct arrayAttr{
    TokenType type;
    char *name;
    int size;
} ArrayAttr;
```

사진_2 배열 선언을 위한 구조체

사진_2와 같이 배열로 변수가 선언되거나, 함수의 인자로 넘어오는 경우를 위해서 구조체를 만들어 준다.

```
/* ExpType is used for type checking */
typedef enum {Void,Integer,Boolean,IntegerArray} ExpType;
```

사진_3 수행될 함수의 타입

사진_3과 같이 수행될 함수들의 타입을 지정해준다.

```c
typedef struct treeNode
   { struct treeNode * child[MAXCHILDREN];
     struct treeNode * sibling;
     int lineno;
     NodeKind nodekind;
     union { StmtKind stmt;
             ExpKind exp;
             DecKind dec;
             ParamKind param;
             TokenType type;
     } kind;
     union { TokenType op;
             TokenType type;
             int val;
             char * name;
             ArrayAttr arr;
             struct ScopeRec * scope;
     } attr;
     ExpType type; /* for type checking of exps */
   } TreeNode;
```

사진_4
syntax tree를 구성하는 노드구조체

Syntax tree를 구성하는 노드 구조체를 사진_4와 같이 선언 해 주고, 그 안에 공용체를 사용하여 메모리의 낭비를 줄인다.

- util.c -

```c
/* Function newDecNode creates a new declation
 * node for syntax tree construction
 */
TreeNode * newDecNode(DecKind kind)
{ TreeNode * t = (TreeNode *) malloc(sizeof(TreeNode));
  int i;
  if (t==NULL)
    fprintf(listing,"Out of memory error at line %d\n",lineno);
  else {
    for (i=0;i<MAXCHILDREN;i++) t->child[i] = NULL;
    t->sibling = NULL;
    t->nodekind = DecK;
    t->kind.dec = kind;
    t->lineno = lineno;
  }
  return t;
}
```

사진_5
해당되는 새로운 state 를 위한 구조체

주어진 문법대로 파싱을 해나가며 해당되는 문법의 종류에 따라서 각각 다른 구조체를 만든다. 사진_5는 함수나 변수를 선언 할때에 생성되는 구조체이다.

이 외에도, newExpode, newParamNode, newTypeNode 등의 (각각 심볼을 표현, 인수들을 위한것, 자료형의 타입 나타냄) 추가적인 구조체들이 선언했다.

또한 사진_6 처럼 print_tree 함수에서는 가각 노드의 현재 상태에 따라서 다른 값들을 출력하게 해주었다. (print_tree 함수는 일부만 발췌했습니다)

```
void printTree( TreeNode * tree )
{ int i;
  INDENT;
  while (tree != NULL) {
    printSpaces();
    if (tree->nodekind==StmtK)
    { switch (tree->kind.stmt) {
        case CompK:
          fprintf(listing,"Compound Statement :\n");
          break;
        case IfK:
          fprintf(listing,"If (condition) (body) (else)\n");
          break;
        case IterK:
          fprintf(listing,"Repeat\n");
          break;
        case RetK:
          fprintf(listing,"Return\n");
          break;
        default:
          fprintf(listing,"Unknown ExpNode kind\n");
          break;
      }
    }
```

사진_6
뒤의 token을 받아올때 덮어쓰기가 되어 날아가는 현상을 방지하기 위한 규칙

- cminus.y -

```
%token IF ELSE WHILE RETURN INT VOID
%token ID NUM
%token ASSIGN EQ NE LT LE GT GE PLUS MINUS TIMES OVER
%token LPAREN RPAREN LBRACE RBRACE LCURLY RCURLY SEMI COMMA
%token ERROR
```

사진_10
token

Cmionus.y에서는 실제 파싱할 경우 어떤식으로 하는지에 관한 Code들이 적혀있다. 우선 사진_10 처럼 token을 선언을 해준 뒤,

```
var_dec     : type_spec saveName SEMI
                { $$ = newDecNode(VarK);
                  $$->child[0] = $1;
                  $$->lineno = lineno;
                  $$->attr.name = savedName;
                }
            | type_spec saveName LBRACE saveNumber RBRACE SEMI
                { $$ = newDecNode(ArrVarK);
                  $$->child[0] = $1;
                  $$->lineno = lineno;
                  $$->attr.arr.name = savedName;
                  $$->attr.arr.size = savedNumber;
                }
            ;
```

사진_11
변수를 선언 할 때 규칙

사진_11 처럼 각 문법의 규칙에 맞는 Code들을 작성 해 주었다.

```
saveName     : ID
                 { savedName = copyString(tokenString);
                   savedLineNo = lineno;
                 }
             ;
saveNumber   : NUM
                 { savedNumber = atoi(tokenString);
                   savedLineNo = lineno;
                 }
             ;
```

사진_12
주어진 규칙에 없는 새로운 규칙

또 마지막 토큰만을 저장하므로, ID를 가져올때에 뒤의 토큰을 처리하는 경우 기존에 읽었던 토큰이 덮어써져 날아가는 현상을 방지하기 위해 사진_12 처럼 새로운 규칙을 추가하였다.

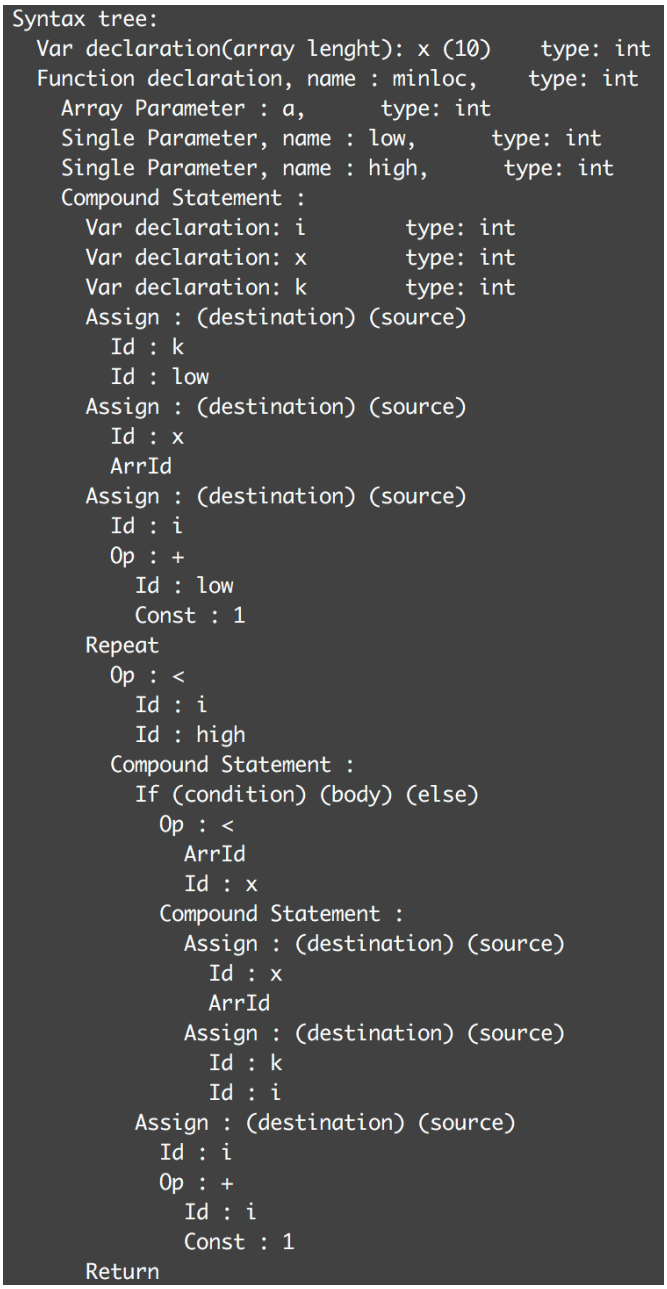
4. 실행결과

```
Syntax tree:
  Function declaration, name : gcd,    type: int
    Single Parameter, name : u,      type: int
    Single Parameter, name : v,      type: int
    Compound Statement :
      If (condition) (body) (else)
        Op : ==
          Id : v
          Const : 0
        Return
          Id : u
        Return
          Call, name: gcd, with arguments below
            Id : v
            Op : -
              Id : u
              Op : *
                Op : /
                  Id : u
                  Id : v
                Id : v
  Function declaration, name : main,    type: void
    type: void
    Compound Statement :
      Var declaration: x        type: int
      Var declaration: y        type: int
      Assign : (destination) (source)
        Id : x
        Call, name: input, with arguments below
      Assign : (destination) (source)
        Id : y
        Call, name: input, with arguments below
      Call, name: output, with arguments below
        Call, name: gcd, with arguments below
          Id : x
          Id : y
```

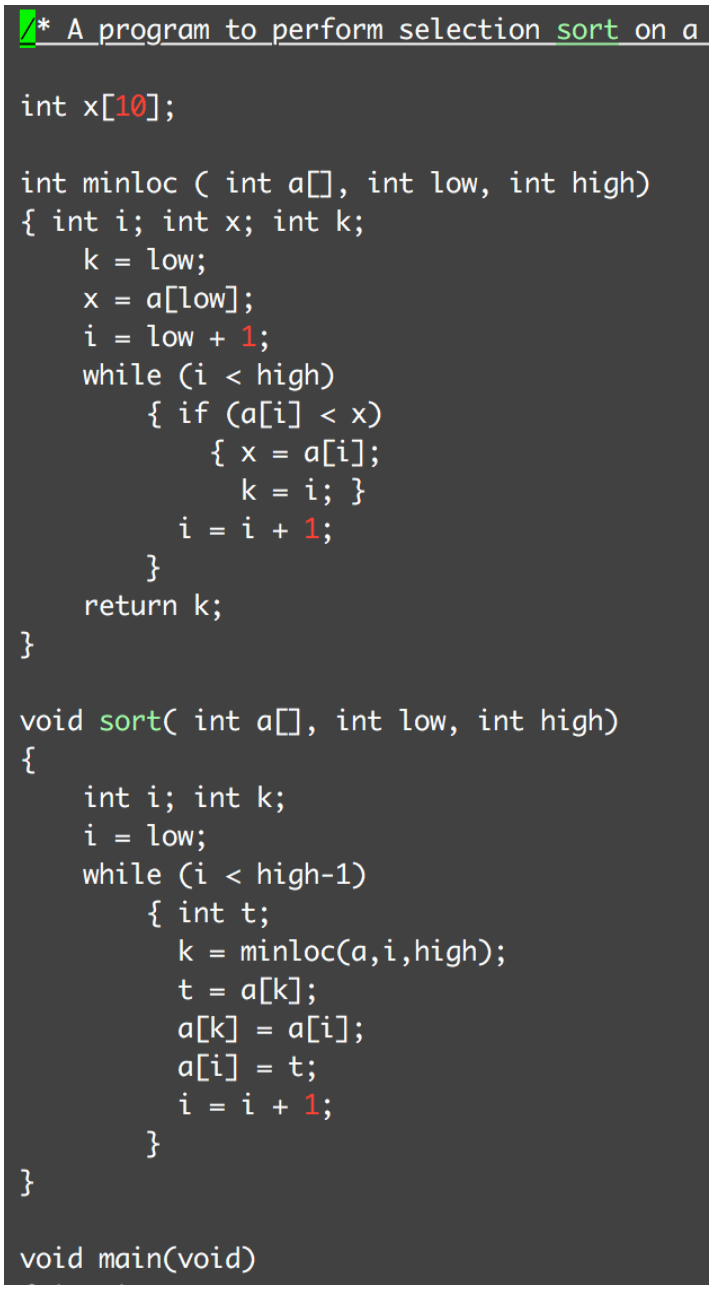
```
/* A program to perform Euclid's
Algorithm to computer gcd */

int gcd (int u, int v)
{
    if (v == 0) return u;
    else return gcd(v,u-u/v*v);
    /* u-u/v*v == u mod v */
}
void main(void)
{
    int x; int y;
    x = input(); y = input();
    output(gcd(x,y));
}
```

사진_13(왼쪽) 사진_14(오른쪽)
실행결과

사진_13은 사진_14의 Code를 받아서 실행한 결과이다. 주어진 명세와 출력화면이 동일하게 파싱을 하고 있지는 않지만, 올바른 파싱을 하는 것을 알 수 있다.



```
Syntax tree:
  Var declaration(array lenght): x (10)    type: int
  Function declaration, name : minloc,    type: int
    Array Parameter : a,      type: int
    Single Parameter, name : low,      type: int
    Single Parameter, name : high,      type: int
    Compound Statement :
      Var declaration: i        type: int
      Var declaration: x        type: int
      Var declaration: k        type: int
      Assign : (destination) (source)
        Id : k
        Id : low
      Assign : (destination) (source)
        Id : x
        ArrId
      Assign : (destination) (source)
        Id : i
        Op : +
          Id : low
          Const : 1
      Repeat
        Op : <
          Id : i
          Id : high
        Compound Statement :
          If (condition) (body) (else)
            Op : <
              ArrId
              Id : x
            Compound Statement :
              Assign : (destination) (source)
                Id : x
                ArrId
              Assign : (destination) (source)
                Id : k
                Id : i
          Assign : (destination) (source)
            Id : i
            Op : +
              Id : i
              Const : 1
      Return
```



```
/* A program to perform selection sort on a

int x[10];

int minloc ( int a[], int low, int high)
{ int i; int x; int k;
    k = low;
    x = a[low];
    i = low + 1;
    while (i < high)
        { if (a[i] < x)
            { x = a[i];
              k = i; }
          i = i + 1;
        }
    return k;
}

void sort( int a[], int low, int high)
{
    int i; int k;
    i = low;
    while (i < high-1)
        { int t;
          k = minloc(a,i,high);
          t = a[k];
          a[k] = a[i];
          a[i] = t;
          i = i + 1;
        }
}

void main(void)
```

사진_15
실행화면

사진_16

사진_16의 Code(일부만 발췌했습니다)를 실행시 사진_15의 결과화면이 나오는 것을 알 수 있으며(일부만 발췌했습니다) 사진_15에서 확인 할 수 있듯이, 배열을 변수로 선언하고 함수의 인자로 넘겨주어도 이를 잘 파싱하는 것을 확인하였다.