

## 컴파일러설계 Project 2(Parser)

2016026599

정보시스템학과 황성우

### 1. Compilation method and environment (컴파일 방법과 개발환경)

컴파일과 실행은 모두 Virtualbox 안에서 Ubuntu 14.04 으로 진행하였습니다. 컴파일은 gcc 5.4.0을 사용하였고 lex는 apt-get 을 이용하여 lex 2.6.0을 사용했습니다. Makefile은 과제 spec에 올려주신 것을 사용하였고, analyze 부분과 cgen 부분은 주석처리하였습니다.

### 2. Explanation about how to implement and how to operate

수정하여 구현한 파일은 크게 globals.h, util.c , cminus.y,makefile로 4가지입니다.

(Makefile은 위에서 설명함)

#### 1) globals.h

yacc 폴더 안에 있는 globals.h를 덮어씌웠고 과제 pdf 안에 있는 힌트를 참고하였습니다. 파스트리 print 할때 필요한 노드를 정의해야해서 treenode struct를 수정하였습니다. 또 추가되는 노드들을 syntax\_node.pdf 파일을 참고하여 세부적인 노드들도 추가해주었습니다.

#### 2) util.c

util.c에서는 파싱을 하면서 나온 결과를 화면에 출력해주는 print 파일이라고 생각했습니다. 기존의 있던 print tree 함수를 수정하면서 위에 적은듯 globals.h에서 추가한 노드타입을 인식하여 프린트할수 있게 수정하였습니다.

#### 3) cminus.y

cminus.y 안에서 cminus의 BNF문법에 맞게 작성하는 메인파일이라고 이해했습니다. cminus의 문법을 적는건 크게 어렵지 않았지만 문법 마다 이루어지는 action code 부분이 가장 어려웠던 것 같습니다. 예를 들어서 *while( t->sibling!=NULL)/* 로 시작하는 문법에서 구조 자체가 이해되지 않았습니다. 그래서 기존에 있던 tiny 문법의 action code를 참고해 수정하였습니다. 기존의 문법에서는 array를 표현하지 못하기 때문에 savedArraySizeNumber라는 전역 변수를 추가하여 array 문법을 표현할수 있게 해주었습니다. 나머지는 BNF 규칙에 맞춘대로 cminus 문법을 추가하였습니다. 가장 어려웠던 파일수정이였습니다 πππ

#### 4) makefile

위에서 간단하게 언급했지만 cminus.y,util.c,globals.h를 수정하고 컴파일을 시도하자 analyze 부분과 cgen 부분에서 오류가 났고, 조교님에게 문의해서 그 두부분을 주석처리해도 상관없다고 하셔서 main이 만들어질때 analyze.o와 cgen.o를 포함하지 않는 형태로 makefile을 수정하였습니다.

#### 5) How to operate?

makefile이 이미 구성되어 있기 때문에 make clean을 먼저 한 후 make를 하면 cminus 파일이 생성됩니다 . 그 후에 ./cminus <target cminus file> 을 입력하면 아래와 같은 식의 결과를 확인 할 수 있습니다.

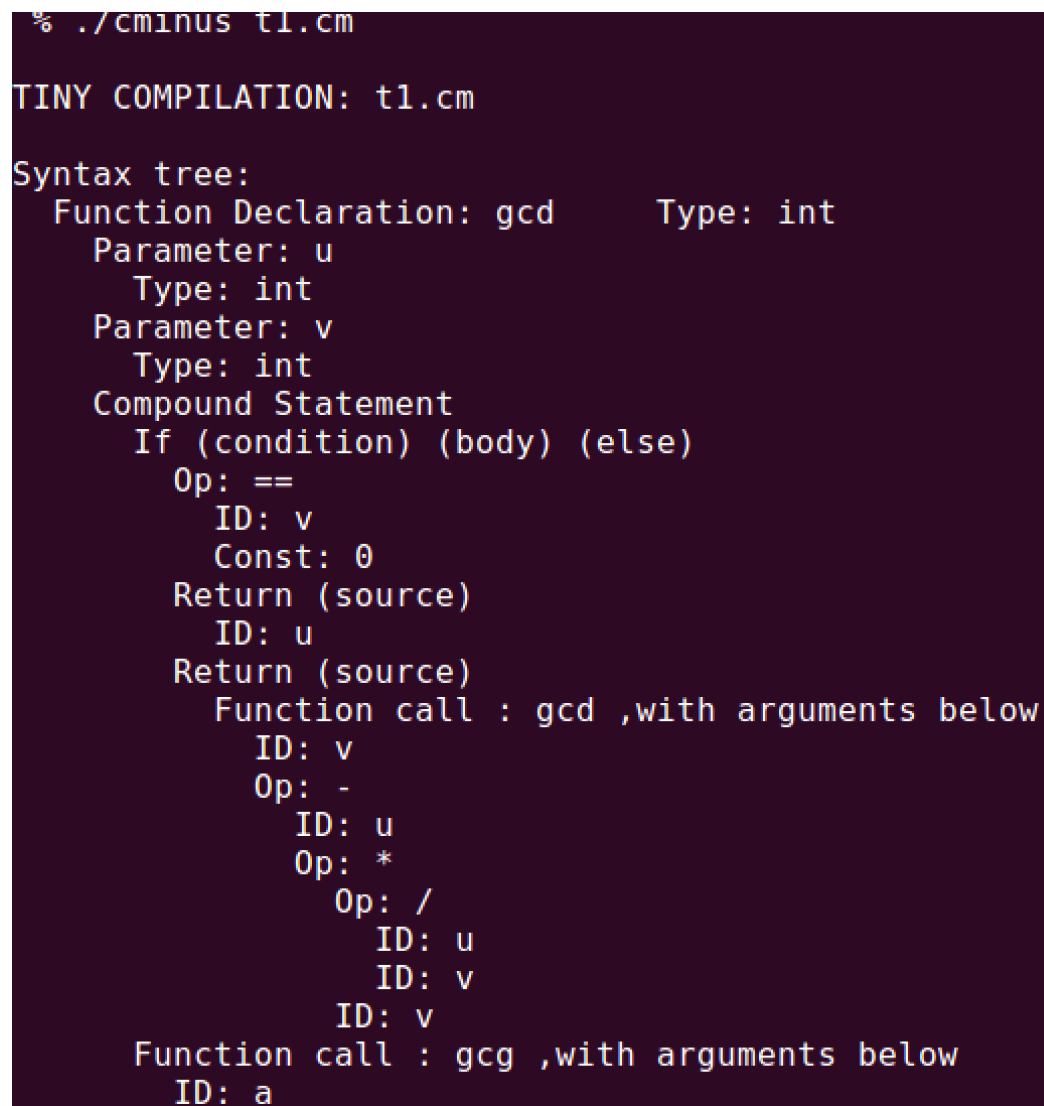
### 3. Example and Result Screenshot

1)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));
}
```

위의 코드를 테스트 해봤습니다.



```
% ./cminus t1.cm

TINY COMPILATION: t1.cm

Syntax tree:
  Function Declaration: gcd      Type: int
    Parameter: u
      Type: int
    Parameter: v
      Type: int
    Compound Statement
      If (condition) (body) (else)
        Op: ==
          ID: v
          Const: 0
        Return (source)
          ID: u
        Return (source)
          Function call : gcd ,with arguments below
            ID: v
            Op: -
              ID: u
              Op: *
                Op: /
                  ID: u
                  ID: v
                ID: v
          Function call : gcg ,with arguments below
            ID: a
```

```
184 a
Function Declaration: main      Type: void
Type: void
Compound Statement
  Variable Declaration: x
  Type: int
  Variable Declaration: y
  Type: int
  Assign to: (destination) (source)
  ID: x
  Function call : input ,with arguments below
  Assign to: (destination) (source)
  ID: y
  Function call : input ,with arguments below
  Function call : output ,with arguments below
  Function call : gcd ,with arguments below
  ID: x
  ID: y
```

정상적으로 파싱하는 것을 확인할 수 있었습니다.