컴파일러 3

Install

1. Dev-C++
   - 관련 사이트 링크: http://www.bloodshed.net/devcpp.html
   - Dev-C++를 설치하면(주의: 이것이 설치되는 폴더(folder)까지 이르는 경로명(path name)에 공백이 없도록 한다. 보통 'C:₩Dev-Cpp'에 설치한다) 그 설치 디렉터리(directory) 아래 'bin'이라는 하위 디렉터리가 만들어지고 이 안에 GNU C 컴파일러('gcc.exe')와 C++ 컴파일러('g++.exe')가 설치됨

2. Flex
   - Flex를 다운로드 받는다(관련 사이트 링크: http://gnuwin32.sourceforge.net/packages/flex.htm)
   - 설치 파일을 실행시킨 후 몇 번 '동의' 또는 '다음' 버튼을 누르기만 하면 됨
   - 주의: Dev-C++와 마찬가지로 이것이 설치되는 폴더까지 이르는 경로명에 공백이 없도록 한다. 보통 'C:₩GnuWin32'에 설치함

Install

3. Bison
   - 관련 사이트 링크: http://gnuwin32.sourceforge.net/packages/bison.htm
   - Flex와 같은 방법으로 설치함
   - 주의: 앞의 두 프로그램과 마찬가지로 이것이 설치되는 폴더까지 이르는 경로명에 공백이 없도록 함
   - 경로명에 공백이 있는 경우 실제 Bison을 사용할 때 오류가 발생하는 것을 확인했으니 특히 주의해야 함
   - 보통 Flex와 함께 'C:₩GnuWin32'에 설치함.

Install

4. 환경변수 설정
   - 세 프로그램의 설치를 마치면 시스템의 환경 변수들 중 'PATH' 변수의 값에 GNU C 컴파일러와 Flex, Bison의 실행 파일이 있는 경로를 추가
   - 예컨대 'C:₩Dev-Cpp₩bin;C:₩GnuWin32₩bin' 을 추가 내지 ('PATH' 변수가 없었던 경우) 새롭게 등록

Install

5. 버전확인

- 명령 프롬프트(command prompt) 창을 연다('모든 프로그램' → '보조 프로그램' → '명령 프롬프트' 메뉴를 선택하여 실행하거나 실행 창에서 'cmd(.exe)'를 입력하여 바로 실행시킨다).
- 이 창에서 'flex --version'을 입력하여 설치된 Flex의 버전을 확인한다(최신 버전은 2.5.4).
- 'bison --version'을 입력하여 설치된 Bison의 버전을 확인한다(최신 버전은 2.4.1).
- 'gcc --version'을 입력하여 설치된 GNU C 컴파일러의 버전을 확인한다(최신 버전은 3.4.2이나 Dev-C++ 안정 버전에 포함된 것은 3.3.1)

Build & Run

Flex 입력 파일의 이름은 'test.l'
Bison 입력 파일의 이름은 'test.y' 라고 가정한다.

```
Dos> flex test.l
Dos> bison test.y
```

Build & Run

Example Code

- Build

```
Dos> flex test.l
Dos> gcc -o test lex.yy.c -lfl
```

- Run

```
Dos> test
```

Build & Run

Example Code

- Mkdir workspace/

```
Dos> cd c:₩GnuWin32
Dos> mkdir workspace
Dos> cd workspace
Dos> mkdir example
Dos> cd example
```

- Run

```
Dos> make clean
Dos> make
```

Simple_cal.y

```
%{
#include <stdio.h>
#include <stdlib.h>
%}
%union {
    double double_val;
}
%token<double_val> NUMBER
%left ADD SUB
%left MUL DIV
%left OPEN CLOSE
%token CR
%type<double_val> expression primary_expression
%%
line_list
    :line
    |line_list line
    ;
line
    :expression CR { printf(" =  %lf₩n", $1); }
    | CR
    ;
```

Simple_cal.y

```
expression
    :primary_expression
    |expression ADD expression { $$ = $1 + $3; }
    |expression SUB expression { $$ = $1 - $3; }
    |expression MUL expression { $$ = $1 * $3; }
    |expression DIV expression { $$ = $1 / $3; }
    ;
primary_expression
    :NUMBER
    | OPEN expression CLOSE { $$ = $2; }
    ;
%%
int yyerror(char const *str) {
    extern char *yytext;
    fprintf(stderr, "parser error near %s₩n", yytext);
    return 0;
}
```

Simple_cal.y

```
int main(void) {
    extern int yyparse(void);
    extern FILE *yyin;

    printf(" ex) 8 * 2 ₩n₩n");

    yyin = stdin;
    if(yyparse()) {
        fprintf(stderr, "Error!₩n");
        exit(1);
    }
}
```

Simple_cal.l

```
%{
#include <stdio.h>
#include "y.tab.h"

int yywrap(void) {
    return 1;
}
%}
%%


"+"  { return ADD; }
"-"  { return SUB; }
"*"  { return MUL; }
"/"  { return DIV; }
"("  { return OPEN; }
")"  { return CLOSE; }
"₩n" { return CR; }

([0-9]+)|([0-9]*"."[0-9]+) {
    double temp;
    sscanf(yytext, "%lf", &temp);
    yylval.double_val = temp;
    return NUMBER;
}
[₩t ]+ ;
. return EOF;
%%
```