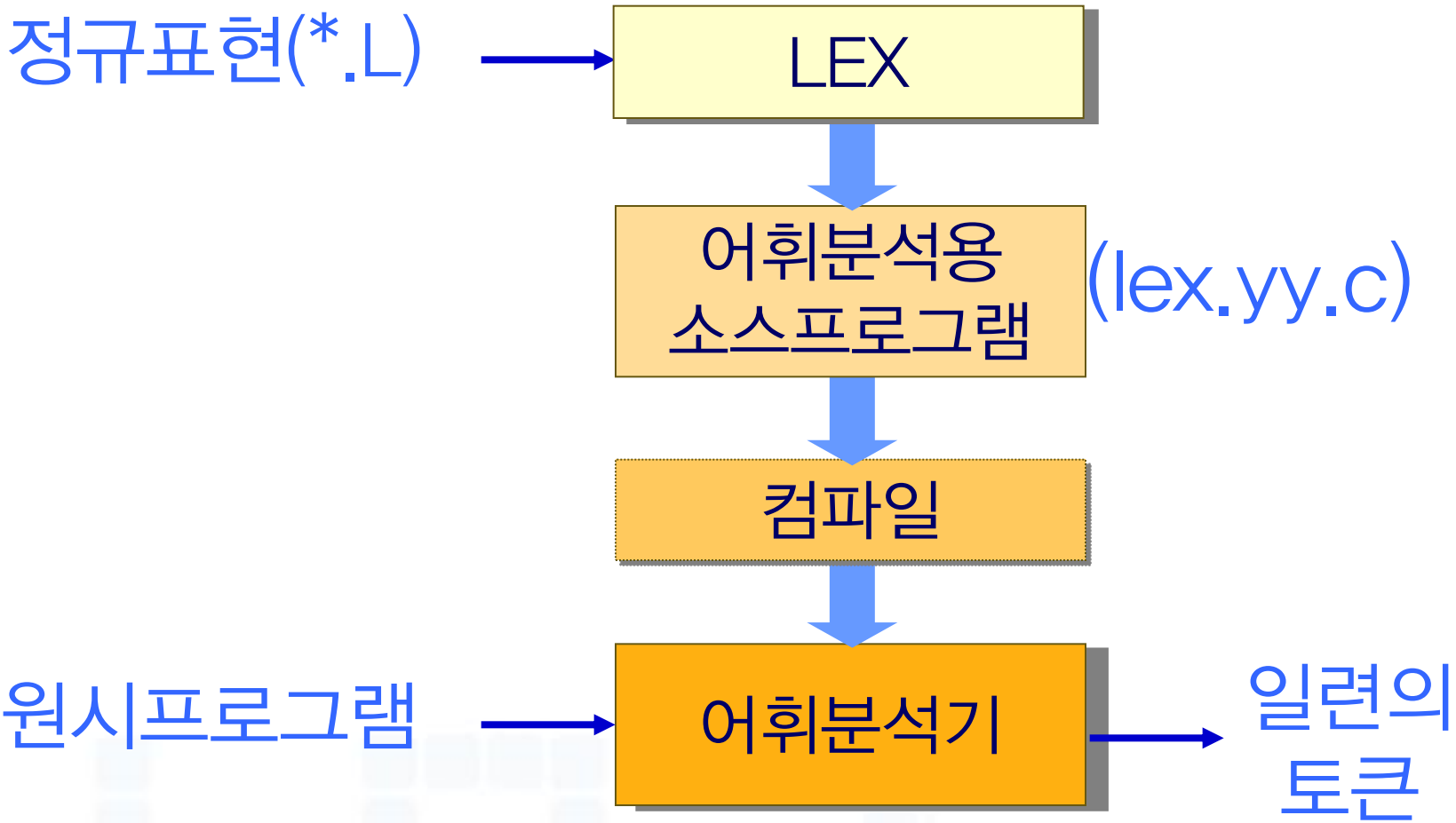
목차

LEX의 기능



사용형식

{ 정의부분 }

%% //규칙시작표시, 생략불가능

{ 규칙부분 } //규칙부분::= 정규표현 + 수행코드

%%

{ 사용자 부프로그램 }

LEX 예제_01

원시프로그램(datafile)

```
ABC := E * 314 + ABC/E;
```

토큰 List(10개)

ABC	+
:=	ABC
E	/
*	E
314	;

정의부분

파일명: TEST.L

```
%{ /* 생성된 프로그램에 그대로 copy됨 */
   #include <stdio.h>
   #include <stdlib.h>
   enum tnumber {TEOF, TIDEN, TNUM,
   TASSIGN, TADD, TPOINT, TMUL, TDIV,
   TSEMI, TDOT, TBEGIN, TEND, TERROR};
%}

/* 이름n          치환식 */
    letter        [a-zA-Z]
    digit         [0-9]
```

규칙부분

파일명: TEST.L

```
%%
    begin                          return(TBEGIN);
    end                            return(TEND);
    {letter}({letter}|{digit})*    return(TIDEN);
    ":="                           return(TASSIGN);
    "+"                            return(TADD);
    "*"                            return(TMUL);
    "/"                            return(TDIV);
    {digit}+                       return(TNUM);
    ";"                            return(TSEMI);
    ₩.                             return(TDOT);
    [ ₩t₩n]                        ;
    .                              return(TERROR);
```

사용자 부프로그램 부분

파일명: TEST.L

```
%%
 main()
 {
 enum    tnumber   tn;      /* token number */
 printf("    Start of   LEX    ₩n");

 while ((tn = yylex() != TEOF) switch (tn) {
        case TBEGIN       : printf("Begin   ₩n"); break;
        case TEND         : printf("End   ₩n"); break;
        case TIDEN        : printf("Identifier : %s₩n", yytext); break;
        case TASSIGN      : printf("Assign_op ₩n"); break;
        case TADD         : printf("Add_op   ₩n"); break;
        case TMUL         : printf("Mul_op   ₩n"); break;
        case TDIV         : printf("Div_op   ₩n"); break;
        case TNUM         : printf("Number: %s₩n", yytext); break;
        case TSEMI        : printf("Semicolon   ₩n"); break;
        case TDOT         : printf("Dot   ₩n"); break;
        case TERROR       : printf("Error   ₩n"); break;
        }
        yywrap() {printf(" End of LEX   ₩=n");   return 1; }
 }
```

문자들의 classes 정의

- − (dash) : 구간 표시.

  [a–z0–9] ⇔ 모든 소문자와 숫자 표시.

- ^ (hat) : negate 또는 complement.

  [^a–zA–Z] ⇔ letter가 아닌 어떤 문자.

- ₩ (backslash: ₩) : escape.

*, +

*, + : 반복 표현.

a* ⇔ a를 0번 이상.

{letter}({letter}|{digit})*

a+ ⇔ a를 1번 이상.

{digit}+

LEX 수행코드

- 정규표현이 매칭되면 즉,
  토큰이 인식되었을 때 실행해야 할
  행동을 C언어로 기술하는 부분.

- 수행코드 구성
  - 전역변수
  - 함 수
  - I/O 루틴(입출력함수)

전역변수

yytext

정규표현과 매칭된 실제 문자열 보관 변수.

[a–z]+ printf(“%s”, yytext);

yyleng

매칭된 문자열의 길이를 저장하는 변수

yytext[yyleng–1]

⇔ 매칭된 스트링의 마지막 문자

LEX 예제_01

원시프로그램(datafile)

```
ABC := E  *  314 + ABC/E;
```

토큰 List(10개)

```
ABC         +
:=          ABC
E           /
 *          E
314         ;
```

정의부분

파일명: TEST.L

```
%{ /* 생성된 프로그램에 그대로 copy됨 */
   #include <stdio.h>
   #include <stdlib.h>
   enum tnumber {TEOF, TIDEN, TNUM,
   TASSIGN, TADD, TPOINT, TMUL, TDIV,
   TSEMI, TDOT, TBEGIN, TEND, TERROR};
%}

/* 이름n        치환식 */
    letter      [a-zA-Z]
    digit       [0-9]
```

규칙부분

파일명: TEST.L

```
%%
  begin                          return(TBEGIN);
  end                            return(TEND);
  {letter}({letter}|{digit})*    return(TIDEN);
  ":="                           return(TASSIGN);
  "+"                            return(TADD);
  "*"                            return(TMUL);
  "/"                            return(TDIV);
  {digit}+                       return(TNUM);
  ";"                            return(TSEMI);
  ₩.                             return(TDOT);
  [ ₩t₩n]                        ;
  .                              return(TERROR);
```

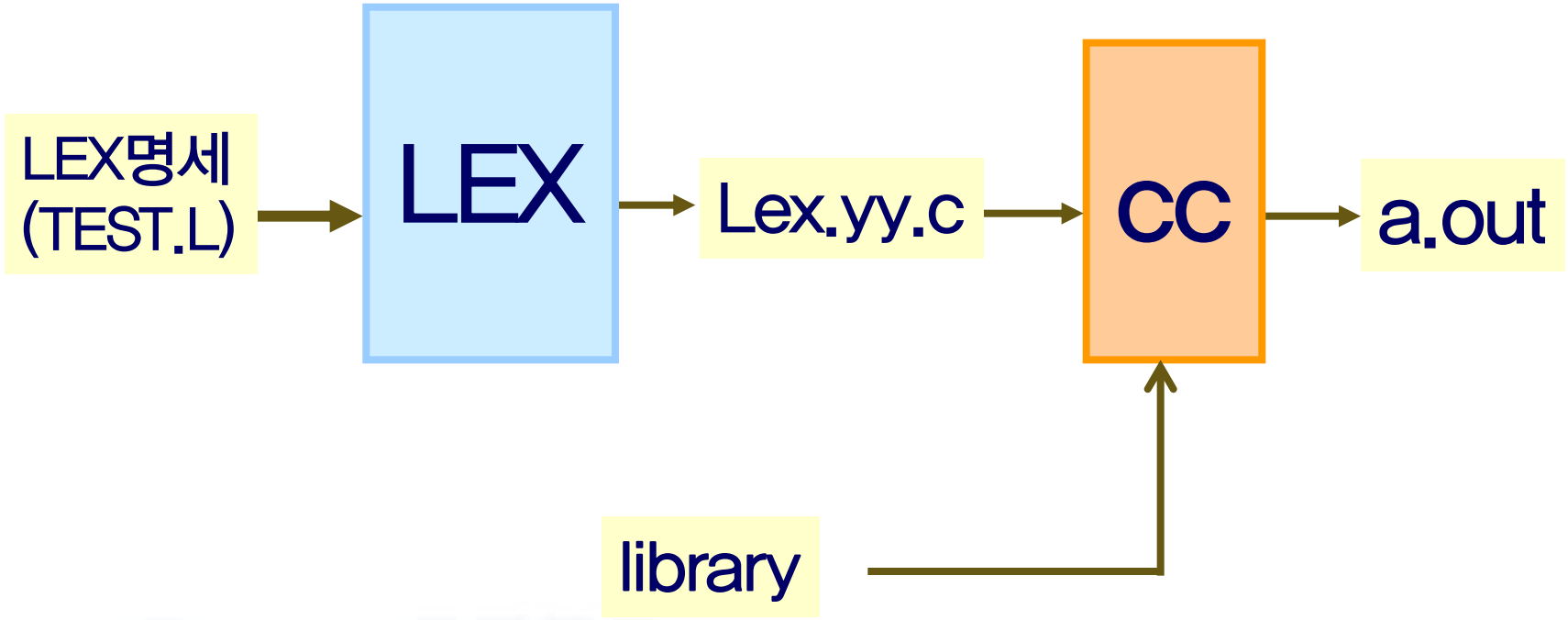
사용자 부프로그램 부분

파일명: TEST.L

```
%%
 main()
 {
 enum    tnumber   tn;      /* token number */
 printf("    Start of   LEX    ₩n");

 while ((tn = yylex() != TEOF) switch (tn) {
        case TBEGIN       : printf("Begin   ₩n"); break;
        case TEND         : printf("End   ₩n"); break;
        case TIDEN        : printf("Identifier : %s₩n", yytext); break;
        case TASSIGN      : printf("Assign_op ₩n"); break;
        case TADD         : printf("Add_op   ₩n"); break;
        case TMUL         : printf("Mul_op   ₩n"); break;
        case TDIV         : printf("Div_op   ₩n"); break;
        case TNUM         : printf("Number: %s₩n", yytext); break;
        case TSEMI        : printf("Semicolon   ₩n"); break;
        case TDOT         : printf("Dot   ₩n"); break;
        case TERROR       : printf("Error   ₩n"); break;
        }
        yywrap() {printf(" End of LEX   ₩=n");   return 1; }
 }
```

실행절차



PC 환경에서 실행법

C 〉 PCL32 –i test.I ↵

C 〉 컴파일(VC++) test.c ↵

C 〉 TEST 〈 data ↵

실행결과


```
Start of LEX
Identifier : ABC
Assign_op
Identifier : E
MUL_op
Number : 314
Add_op
Identifier : ABC
DIV_op
Identifier : E
Semicolon
End of LEX
```

LEX 예제_02

원시프로그램(datafile)

This is a simple word counting program.

단어와 문자 줄의 수 계산

This is a simple word counting program.

- 단어 7
- 문자 39
- 줄 1

정의부분

파일명:wordcount.L

```
%{
unsigned charCount =0, wordCount = 0,
   lineCount = 1;
%}
word  [^  ₩t₩n]+
eol     ₩n
```

규칙부분

파일명:wordcount.L

```
%%
{word}
        {wordCount++; charCount +=yyleng;}
{eol}     {charCount++; lineCount++;}
.       charCount++;
```

사용자 부프로그램 부분

파일명:wordcount.L

```
%%
main()
{
    yylex();
    printf("%d %d %d₩n", lineCount, wordCount, charCount);
}
```

LEX 실습 과정 보기

질문코너1

JLEX

- A lexical analyzer generator for Java(TM)
- Elliot Berk

  Department of Computer Science, Princeton University

- Version 1.2, May 5, 1997
- Manual revision October 29, 1997
- Last updated September 6, 2000 for JLex1.2.5
- (latest version can be obtained from http://www.cs.princeton.edu/~appel/modern/java/JLex/)

❖http://www.abxsoft.com/

제6강의 학습내용

- Context-free 문법의 효율화
- 유도트리와 모호성
- 불필요한 생성규칙
- ε생성규칙 / 단일생성규칙
- Left-factoring / Left-recursion
- Push-Down Automata

수고하셨습니다.

제6강에서 만나요~