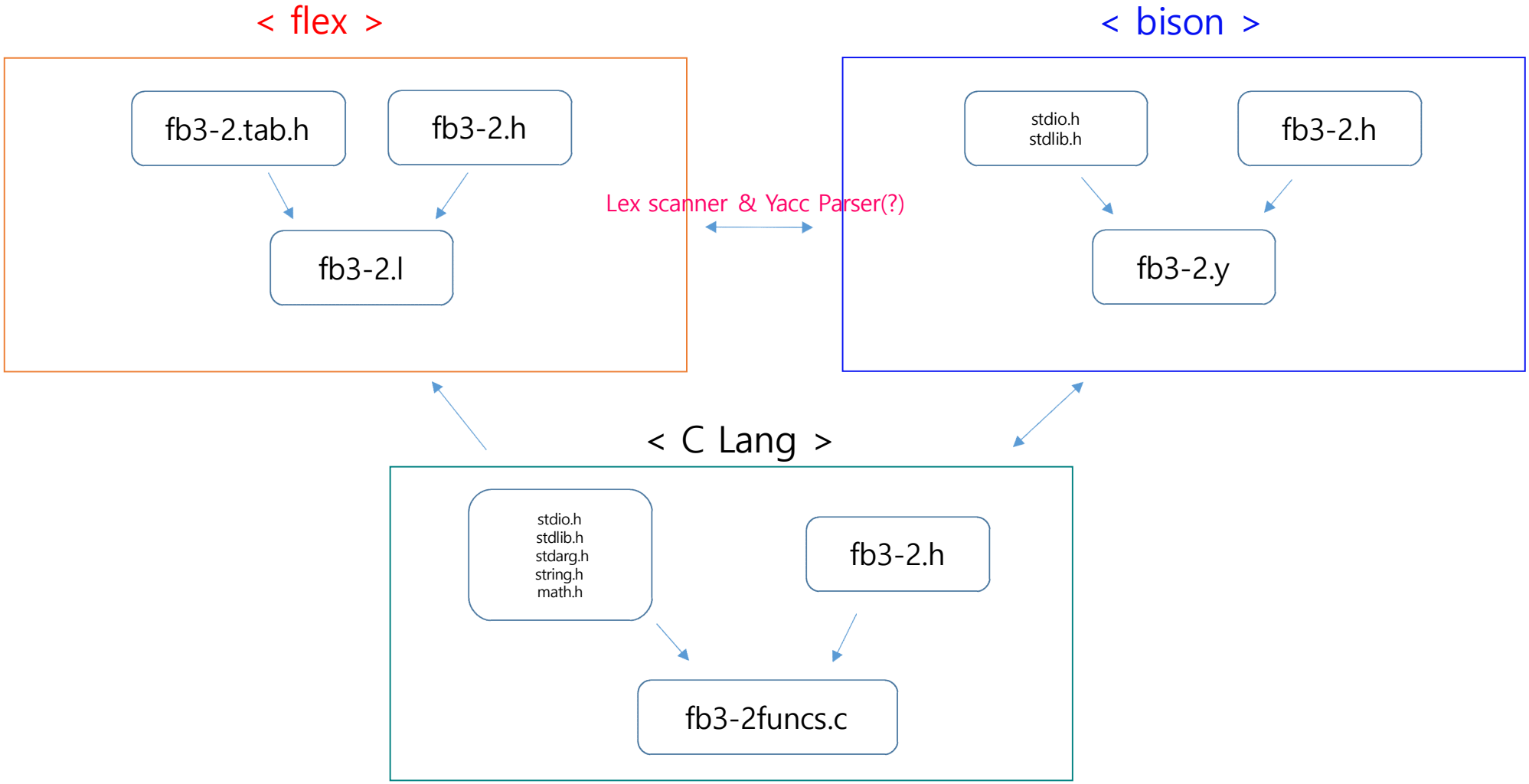
Anti-Virus Homework #1

서재언

- Flex & Bison 실행 구조

How to Transfer tokens from flex to bison ?
Please, Explain this procedure, Mento :)



- 사용자 변수 및 함수 선언(fb3-2.h)

```
/* declare implicit function */
int yyparse(void);
int yylex(void);



/* symbol table */
struct symbol {                         /* a variable name */
  char *name;
  double value;
  struct ast *func;             /* stmt for the function */
  struct symlist *syms; /* list of dummy args */
};



/* list of symbols, for an argument list */
struct symlist {
  struct symbol *sym;
  struct symlist *next;
};

struct symlist *newsymlist(struct symbol *sym, struct symlist *next);
void symlistfree(struct symlist *sl);
```

- 사용자 변수 및 함수 선언(fb3-2.h)

```
enum bifs {                          /* built-in functions */
  B_sqrt = 1,
  B_exp,
  B_log,
  B_print
};

/* nodes in the Abstract Syntax Tree */
/* all have common initial nodetype */

struct ast {
  int nodetype;
  struct ast *l;
  struct ast *r;
};

struct fncall {                      /* built-in function */
  int nodetype;                              /* type F */
  struct ast *l;
  enum bifs functype;
};

struct ufncall {                     /* user function */
  int nodetype;                              /* type C */
  struct ast *l;                     /* list of arguments */
  struct symbol *s;
};
```

```
/* node types
 *  + - * / |
 *  0-7 comparison ops, bit coded 04 equal, 02 less, 01 greater
 *  M unary minus
 *  L statement list
 *  I IF statement
 *  W WHILE statement
 *  N symbol ref
 *  = assignment
 *  S list of symbols
 *  F built in function call
 *  C user function call
 */
```

- 사용자 변수 및 함수 선언(fb3-2.h)

```
struct flow {
  int nodetype;                 /* type I or W */
  struct ast *cond;             /* condition */
  struct ast *tl;               /* then or do list */
  struct ast *el;               /* optional else list */
};


struct numval {
  int nodetype;                         /* type K */
  double number;
};


struct symref {
  int nodetype;                         /* type N */
  struct symbol *s;
};


struct symasgn {
  int nodetype;                         /* type = */
  struct symbol *s;
  struct ast *v;                        /* value */
};
```

```
/* node types
 *  + - * / |
 *  0-7 comparison ops, bit coded 04 equal, 02 less, 01 greater
 *  M unary minus
 *  L statement list
 *  I IF statement
 *  W WHILE statement
 *  N symbol ref
 *  = assignment
 *  S list of symbols
 *  F built in function call
 *  C user function call
 */
```

- 사용자 변수 및 함수 선언(fb3-2.h)

```
/* build an AST */
struct ast *newast(int nodetype, struct ast *l, struct ast *r);
struct ast *newcmp(int cmptype, struct ast *l, struct ast *r);
struct ast *newfunc(int functype, struct ast *l);
struct ast *newcall(struct symbol *s, struct ast *l);
struct ast *newref(struct symbol *s);
struct ast *newasgn(struct symbol *s, struct ast *v);
struct ast *newnum(double d);
struct ast *newflow(int nodetype, struct ast *cond, struct ast *tl, struct ast *tr);



/* define a function */
void dodef(struct symbol *name, struct symlist *syms, struct ast *stmts);

/* evaluate an AST */
double eval(struct ast *);

/* delete and free an AST */
void treefree(struct ast *);

/* interface to the lexer */
extern int yylineno; /* from lexer */
void yyerror(char *s, ...);

extern int debug;
void dumpast(struct ast *a, int level);
```

- 사용자 변수 및 함수 정의(fb3-2.func.c)

struct symbol symtab[NHASH];

- symbol 구조체 배열을 전역으로 설정

```
/* simple symtab of fixed size */
#define NHASH 9997
```

static unsigned symhash(char *sym)

- hash = hash*9 ^ [1byte char] 를 통해 hash값을 생성하는 함수

struct symbol *lookup(char* sym)

- 인자 sym 문자열이 symtab 구조체에 저장되어 있는지 찾아보고 없다면 새로운 entry 추가

struct ast *newast(int nodetype, struct ast *l, struct ast *r)

- AST에 nodetype과 L, R 노드 추가

struct ast *newnum(double d)

- AST에 nodetype=K에 해당하는 constant value d 저장 및 추가

struct ast *newcmp(int cmptype, struct ast *l, struct ast *r)

- AST에서 comtype을 통해 L,R의 값을 비교(gt, lt, eq 등..)

struct ast *newfunc(int functype, struct ast *l)

- built in function call 추가

- 사용자 변수 및 함수 정의(fb3-2.func.c)

struct ast *newcall(struct symbol *s, struct ast *l)

- user function call 추가

struct ast *newref(struct symbol *s)

- symbol ref 추가

struct ast *newasgn(struct symbol *s, struct ast *v)

- symbol에 값을 할당

struct ast *newflow(int nodetype, struct ast *cond, struct ast *tl, struct ast *el)

- 조건문(If ~ then, If ~ then ~ Else, While)을 생성하는 노드 생성

struct symlist *newsymlist(struct symbol *sym, struct symlist *next)

- list에 새로운 sym 구조체 추가

void symlistfree(struct symlist *sl)

- list에서 해당 sl 노드 메모리 해제

void dodef(struct symbol *name, struct symlist *syms, struct ast *func)

- symbol 구조체에 list syms와 func 추가

- 사용자 변수 및 함수 정의(fb3-2.func.c)

double eval(struct ast *a)

- nodetype에 따른 AST Travel 및 연산

static double callbuiltin(struct fncall *f)

- builtin-funcion(sqrt, exp, log, print) 호출

static double calluser(struct ufncall *f)

- AST Travel을 통한 user-funcion 호출

void treefree(struct ast *a)

- 해당 AST 노드 메모리 해제

void yyerror(char *s, ...)

- 가변인자를 통한 에러 출력

int main()

- main()함수에서 yyparse()함수는 yacc에 의해 만들어지는 구문분석기를 호출
- yyparse()함수는 yylex()라는 lex가 만들어 주는 해석기(lexer)를 이용해 입력열에 대한 토큰을 처리

- 계산기 프로그램 실행 결과

fb3-2.exe	2023-05-20 오후 8:04	응용 프로그램	88KB
fb3-2.h	2023-05-20 오후 8:04	C/C++ Header	3KB
fb3-2.l	2009-11-07 오후 9:54	L 파일	2KB
fb3-2.lex.c	2023-05-20 오후 8:04	C Source	51KB
fb3-2.tab.c	2023-05-20 오후 8:04	C Source	48KB
fb3-2.tab.h	2023-05-20 오후 8:04	C/C++ Header	4KB
fb3-2.y	2009-11-07 오후 9:54	Y 파일	3KB
fb3-2funcs.c	2023-05-20 오후 9:48	C Source	11KB
Makefile	2023-05-20 오후 7:58	파일	1KB