```
 1: ::::::::::::::
 2: absyn.mli.defs
 3: ::::::::::::::
 4: type linenr = int
 5: type ident = string
 6: type label = string
 7: type number = float
 8: type oper = string
 9: and memref = Arrayref of ident * expr | Variable of ident
10: and expr =
11:     Number of number
12:   | Memref of memref
13:   | Unary of oper * expr
14:   | Binary of oper * expr * expr
15: type printable = Printexpr of expr | String of string
16: type stmt =
17:     Dim of ident * expr
18:   | Let of memref * expr
19:   | Goto of label
20:   | If of expr * label
21:   | Print of printable list
22:   | Input of memref list
23: type progline = linenr * label option * stmt option
24: type program = progline list
```

```
 1: ::::::::::::::
 2: dumper.mli.defs
 3: ::::::::::::::
 4: val quote : string -> string
 5: val join : string -> string -> string -> string list -> string
 6: val string_of_option : ('a -> string) -> 'a option -> string
 7: val string_of_ctor : string -> string list -> string
 8: val string_of_list : ('a -> string) -> 'a list -> string
 9: val string_of_printable : Absyn.printable -> string
10: val string_of_memref : Absyn.memref -> string
11: val string_of_expr : Absyn.expr -> string
12: val string_of_stmt : Absyn.stmt -> string
13: val dump_progline : int * string option * Absyn.stmt option -> unit
14: val dump_program : Absyn.program -> unit
15: ::::::::::::::
16: dumper.ml.defs
17: ::::::::::::::
18: val quote : string -> string
19: val join : string -> string -> string -> string list -> string
20: val string_of_option : ('a -> string) -> 'a option -> string
21: val string_of_ctor : string -> string list -> string
22: val string_of_list : ('a -> string) -> 'a list -> string
23: val string_of_printable : Absyn.printable -> string
24: val string_of_memref : Absyn.memref -> string
25: val string_of_expr : Absyn.expr -> string
26: val string_of_stmt : Absyn.stmt -> string
27: val dump_progline : int * string option * Absyn.stmt option -> unit
28: val dump_program : Absyn.program -> unit
```

```
  1: ::::::::::::::
  2: etc.mli.defs
  3: ::::::::::::::
  4: val warn : string list -> unit
  5: val die : string list -> unit
  6: val syntax_error : Lexing.position -> string list -> unit
  7: val usage_exit : string list -> unit
  8: val read_number : unit -> float
  9: ::::::::::::::
 10: etc.ml.defs
 11: ::::::::::::::
 12: val execname : string
 13: val exit_status_ref : int ref
 14: val quit : unit -> unit
 15: val eprint_list : string list -> unit
 16: val warn : string list -> unit
 17: val die : string list -> unit
 18: val syntax_error : Lexing.position -> string list -> unit
 19: val usage_exit : string list -> unit
 20: val buffer : string list ref
 21: val read_number : unit -> float
```

```
 1: ::::::::::::::
 2: interp.mli.defs
 3: ::::::::::::::
 4: val want_dump : bool ref
 5: val interpret_program : Absyn.program -> unit
 6: ::::::::::::::
 7: interp.ml.defs
 8: ::::::::::::::
 9: exception Unimplemented of string
10: val no_expr : string -> 'a
11: val no_stmt : string -> 'a -> 'b
12: val want_dump : bool ref
13: val eval_expr : Absyn.expr -> float
14: val interpret : Absyn.program -> unit
15: val interp_stmt : Absyn.stmt -> Absyn.program -> unit
16: val interp_print : Absyn.printable list -> Absyn.program -> unit
17: val interp_input : Absyn.memref list -> Absyn.program -> unit
18: val interpret_program : Absyn.program -> unit
```

```
1: ::::::::::::::
2: main.ml.defs
3: ::::::::::::::
4: val interpret_source : string -> unit
```

```
:::::::::::::::
parser.mli.defs
:::::::::::::::
type token =
    RELOP of string
  | EQUAL of string
  | ADDOP of string
  | MULOP of string
  | POWOP of string
  | IDENT of string
  | NUMBER of string
  | STRING of string
  | COLON
  | COMMA
  | LPAR
  | RPAR
  | LSUB
  | RSUB
  | EOL
  | EOF
  | DIM
  | LET
  | GOTO
  | IF
  | PRINT
  | INPUT
val program : (Lexing.lexbuf -> token) -> Lexing.lexbuf -> Absyn.program
:::::::::::::::
parser.ml.defs
:::::::::::::::
type token =
    RELOP of string
  | EQUAL of string
  | ADDOP of string
  | MULOP of string
  | POWOP of string
  | IDENT of string
  | NUMBER of string
  | STRING of string
  | COLON
  | COMMA
  | LPAR
  | RPAR
  | LSUB
  | RSUB
  | EOL
  | EOF
  | DIM
  | LET
  | GOTO
  | IF
  | PRINT
  | INPUT
val linenr : unit -> int
val syntax : unit -> unit
val yytransl_const : int array
val yytransl_block : int array
val yylhs : string
```

```
val yylen : string
val yydefred : string
val yydgoto : string
val yysindex : string
val yyrindex : string
val yygindex : string
val yytablesize : int
val yytable : string
val yycheck : string
val yynames_const : string
val yynames_block : string
val yyact : (Parsing.parser_env -> Obj.t) array
val yytables : Parsing.parse_tables
val program : (Lexing.lexbuf -> token) -> Lexing.lexbuf -> Absyn.program
```

```
1: ::::::::::::::
2: scanner.ml.defs
3: ::::::::::::::
4: val lexerror : Lexing.lexbuf -> unit
5: val newline : Lexing.lexbuf -> unit
6: val lexeme : Lexing.lexbuf -> string
7: val __ocaml_lex_tables : Lexing.lex_tables
8: val token : Lexing.lexbuf -> Parser.token
9: val __ocaml_lex_token_rec : Lexing.lexbuf -> int -> Parser.token
```

```
 1: ::::::::::::::
 2: tables.mli.defs
 3: ::::::::::::::
 4: type variable_table_t = (string, float) Hashtbl.t
 5: type array_table_t = (string, float array) Hashtbl.t
 6: type unary_fn_table_t = (string, float -> float) Hashtbl.t
 7: type binary_fn_table_t = (string, float -> float -> float) Hashtbl.t
 8: type label_table_t = (string, Absyn.program) Hashtbl.t
 9: val variable_table : variable_table_t
10: val array_table : array_table_t
11: val unary_fn_table : unary_fn_table_t
12: val binary_fn_table : binary_fn_table_t
13: val label_table : label_table_t
14: val init_label_table : Absyn.program -> unit
15: val dump_label_table : unit -> unit
16: ::::::::::::::
17: tables.ml.defs
18: ::::::::::::::
19: type variable_table_t = (string, float) Hashtbl.t
20: type array_table_t = (string, float array) Hashtbl.t
21: type unary_fn_table_t = (string, float -> float) Hashtbl.t
22: type binary_fn_table_t = (string, float -> float -> float) Hashtbl.t
23: type label_table_t = (string, Absyn.program) Hashtbl.t
24: val variable_table : variable_table_t
25: val array_table : array_table_t
26: val unary_fn_table : unary_fn_table_t
27: val binary_fn_table : binary_fn_table_t
28: val label_table : label_table_t
29: val init_label_table : Absyn.program -> unit
30: val dump_label_table : unit -> unit
```