1 ) BNF 구문

<translation\_unit> -> <external\_decl> | <translation\_unit> <external\_decl>

<external\_decl> -> <function\_definition> | <decl>

<function\_definition> -> <decl\_specs> <declarator> <decl\_list> <compound\_stat>

| <declarator> <decl\_list> <compound\_stat>

| <decl\_specs> <declarator> <compound\_stat>

| <declarator> <compound\_stat>

<decl> -> <decl\_specs> <init\_declarator\_list> ';'

| <decl\_specs> ';'

<decl\_list> -> <decl>

| <decl\_list> <decl>

<decl\_specs> -> <storage\_class\_spec> <decl\_specs>

| <storage\_class\_spec>

| <type\_spec> <decl\_specs>

| <type\_spec>

| <type\_qualifier> <decl\_specs>

| <type\_qualifier>

<storage\_class\_spec> -> 'auto' | 'register' | 'static' | 'extern' | 'typedef'

<type\_spec> -> 'void' | 'char' | 'short' | 'int' | 'long' | 'float'

| 'double' | 'signed' | 'unsigned'

| <struct\_or\_union\_spec>

| <enum\_spec>

| <typedef\_name>

<type\_qualifier> -> 'const' | 'volatile'

<struct\_or\_union\_spec> -> <struct\_or\_union> <id> '{' <struct\_decl\_list> '}'

| <struct\_or\_union> '{' <struct\_decl\_list> '}'

| <struct\_or\_union> <id>

<struct\_or\_union> -> 'struct' | 'union'

<struct\_decl\_list> -> <struct\_decl> | <struct\_decl\_list> <struct\_decl>

<init\_declarator\_list> -> <init\_declarator>

| <init\_declarator\_list> ',' <init\_declarator>

<init\_declarator> -> <declarator>

| <declarator> '=' <initializer>

<struct\_decl> -> <spec\_qualifier\_list> <struct\_declarator\_list> ';'

<spec\_qualifier\_list> -> <type\_spec> <spec\_qualifier\_list>

| <type\_spec>

| <type\_qualifier> <spec\_qualifier\_list>

| <type\_qualifier>

<struct\_declarator\_list> -> <struct\_declarator>

| <struct\_declarator\_list> ',' <struct\_declarator>

<struct\_declarator> -> <declarator>

| <declarator> ':' <const\_exp>

| ':' <const\_exp>

<enum\_spec> -> 'enum' <id> '{' <enumerator\_list> '}'

| 'enum' '{' <enumerator\_list> '}'

| 'enum' <id>

<enumerator\_list> -> <enumerator>

| <enumerator\_list> ',' <enumerator>

<enumerator> -> <id>

| <id> '=' <const\_exp>

<declarator> -> <pointer> <direct\_declarator>

| <direct\_declarator>

<direct\_declarator> -> <id>

| '(' <declarator> ')'

| <direct\_declarator> '[' <const\_exp> ']'

| <direct\_declarator> '[' ']'

| <direct\_declarator> '(' <param\_type\_list> ')'

| <direct\_declarator> '(' <id\_list> ')'

| <direct\_declarator> '(' ')'

<pointer> -> '\*' <type\_qualifier\_list>

| '\*'

| '\*' <type\_qualifier\_list> <pointer>

| '\*' <pointer>

<type\_qualifier\_list> -> <type\_qualifier>

| <type\_qualifier\_list> <type\_qualifier>

<param\_type\_list> -> <param\_list>

| <param\_list> ',' '...'

<param\_list> -> <param\_decl>

| <param\_list> ',' <param\_decl>

<param\_decl> -> <decl\_specs> <declarator>

| <decl\_specs> <abstract\_declarator>

| <decl\_specs>

<id\_list> -> <id>

| <id\_list> ',' <id>

<initializer> -> <assignment\_exp>

| '{' <initializer\_list> '}'

| '{' <initializer\_list> ',' '}'

<initializer\_list> -> <initializer>

| <initializer\_list> ',' <initializer>

<type\_name> -> <spec\_qualifier\_list> <abstract\_declarator>

| <spec\_qualifier\_list>

<abstract\_declarator> -> <pointer>

| <pointer> <direct\_abstract\_declarator>

| <direct\_abstract\_declarator>

<direct\_abstract\_declarator>-> '(' <abstract\_declarator> ')'

| <direct\_abstract\_declarator> '[' <const\_exp> ']'

| '[' <const\_exp> ']'

| <direct\_abstract\_declarator> '[' ']'

| '[' ']'

| <direct\_abstract\_declarator> '(' <param\_type\_list> ')'

| '(' <param\_type\_list> ')'

| <direct\_abstract\_declarator> '(' ')'

| '(' ')'

<typedef\_name> -> <id>

<stat> -> <labeled\_stat>

| <exp\_stat>

| <compound\_stat>

| <selection\_stat>

| <iteration\_stat>

| <jump\_stat>

<labeled\_stat> -> <id> ':' <stat>

| 'case' <const\_exp> ':' <stat>

| 'default' ':' <stat>

<exp\_stat> -> <exp> ';'

| ';'

<compound\_stat> -> '{' <decl\_list> <stat\_list> '}'

| '{' <stat\_list> '}'

| '{' <decl\_list> '}'

| '{' '}'

<stat\_list> -> <stat>

| <stat\_list> <stat>

<selection\_stat> -> 'if' '(' <exp> ')' <stat>

| 'if' '(' <exp> ')' <stat> 'else' <stat>

| 'switch' '(' <exp> ')' <stat>

<iteration\_stat> -> 'while' '(' <exp> ')' <stat>

| 'do' <stat> 'while' '(' <exp> ')' ';'

| 'for' '(' <exp> ';' <exp> ';' <exp> ')' <stat>

| 'for' '(' <exp> ';' <exp> ';' ')' <stat>

| 'for' '(' <exp> ';' ';' <exp> ')' <stat>

| 'for' '(' <exp> ';' ';' ')' <stat>

| 'for' '(' ';' <exp> ';' <exp> ')' <stat>

| 'for' '(' ';' <exp> ';' ')' <stat>

| 'for' '(' ';' ';' <exp> ')' <stat>

| 'for' '(' ';' ';' ')' <stat>

<jump\_stat> -> 'goto' <id> ';'

| 'continue' ';'

| 'break' ';'

| 'return' <exp> ';'

| 'return' ';'

<exp> -> <assignment\_exp>

| <exp> ',' <assignment\_exp>

<assignment\_exp> -> <conditional\_exp>

| <unary\_exp> <assignment\_operator> <assignment\_exp>

<assignment\_operator> -> '=' | '\*=' | '/=' | '%=' | '+=' | '-=' | '<<='

| '>>=' | '&=' | '^=' | '|='

<conditional\_exp> -> <logical\_or\_exp>

| <logical\_or\_exp> '?' <exp> ':' <conditional\_exp>

<const\_exp> -> <conditional\_exp>

<logical\_or\_exp> -> <logical\_and\_exp>

| <logical\_or\_exp> '||' <logical\_and\_exp>

<logical\_and\_exp> -> <inclusive\_or\_exp>

| <logical\_and\_exp> '&&' <inclusive\_or\_exp>

<inclusive\_or\_exp> -> <exclusive\_or\_exp>

| <inclusive\_or\_exp> '|' <exclusive\_or\_exp>

<exclusive\_or\_exp> -> <and\_exp>

| <exclusive\_or\_exp> '^' <and\_exp>

<and\_exp> -> <equality\_exp>

| <and\_exp> '&' <equality\_exp>

<equality\_exp> -> <relational\_exp>

| <equality\_exp> '==' <relational\_exp>

| <equality\_exp> '!=' <relational\_exp>

<relational\_exp> -> <shift\_expression>

| <relational\_exp> '<' <shift\_expression>

| <relational\_exp> '>' <shift\_expression>

| <relational\_exp> '<=' <shift\_expression>

| <relational\_exp> '>=' <shift\_expression>

<shift\_expression> -> <additive\_exp>

| <shift\_expression> '<<' <additive\_exp>

| <shift\_expression> '>>' <additive\_exp>

<additive\_exp> -> <mult\_exp>

| <additive\_exp> '+' <mult\_exp>

| <additive\_exp> '-' <mult\_exp>

<mult\_exp > -> <cast\_exp>

| <mult\_exp> '\*' <cast\_exp>

| <mult\_exp> '/' <cast\_exp>

| <mult\_exp> '%' <cast\_exp>

<cast\_exp > -> <unary\_exp>

| '(' <type\_name> ')' <cast\_exp>

<unary\_exp> -> <postfix\_exp>

| '++' <unary\_exp>

| '--' <unary\_exp>

| <unary\_operator> <cast\_exp>

| 'sizeof' <unary\_exp>

| 'sizeof' '(' <type\_name> ')'

<unary\_operator> -> '&' | '\*' | '+' | '-' | '~' | '!'

<postfix\_exp> -> <primary\_exp>

| <postfix\_exp> '[' <exp> ']'

| <postfix\_exp> '(' <argument\_exp\_list> ')'

| <postfix\_exp> '(' ')'

| <postfix\_exp> '.' <id>

| <postfix\_exp> '->' <id>

| <postfix\_exp> '++'

| <postfix\_exp> '--'

<primary\_exp> -> <id>

| <const>

| <string>

| '(' exp ')'

<argument\_exp\_list> -> <assignment\_exp>

| <argument\_exp\_list> ',' <assignment\_exp>

<const> -> <int\_const>

| <char\_const>

| <float\_const>

| <enumeration\_const>