while_stat

while (expr) inner_stat

if_stat

if (expr) inner_stat else inner_stat

for_stat

for (expr ; expr ; expr) inner_stat

return_stat

return expr ;

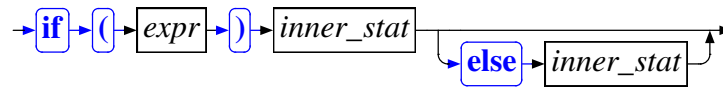*compound_stat*
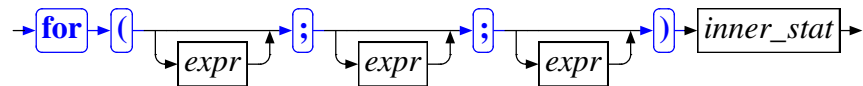
{ stat }

simple_stat

expr ;

decl_stat

dcl simple_decl

empty_stat

;

stat

decl_stat
while_stat
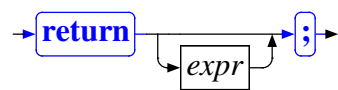if_stat
for_stat
return_stat
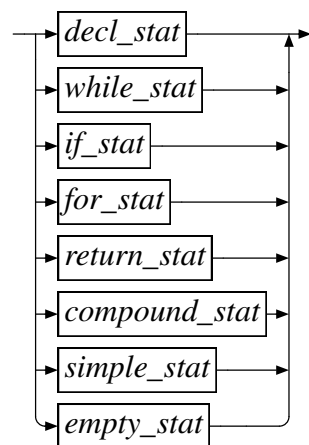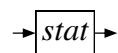compound_stat
simple_stat
empty_stat
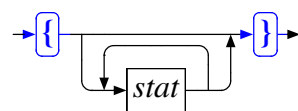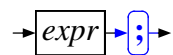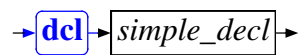
inner_stat

stat

variable_expr

→[**identifier**]→

int_value_expr

→[**number**]→

real_value_expr

→[**real_number**]→

char_value_expr

→[**character_literal**]→

string_value_expr

→[**string_literal**]→

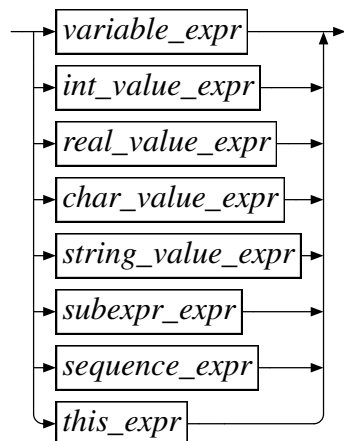*subexpr_expr*

→(**(**)→[*expr*]→(**)**)→

sequence_expr

→(**[**)→[*expr_list*]→(**]**)→

this_expr

→(**this**)→

simple_expr

→[*variable_expr*]→
→[*int_value_expr*]→
→[*real_value_expr*]→
→[*char_value_expr*]→
→[*string_value_expr*]→
→[*subexpr_expr*]→
→[*sequence_expr*]→
→[*this_expr*]→

postfix_expr
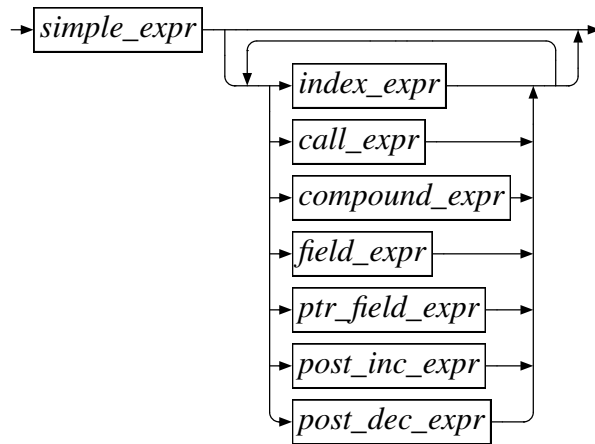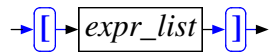
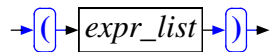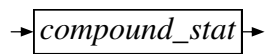*simple_expr* → *index_expr* *call_expr* *compound_expr* *field_expr* *ptr_field_expr* *post_inc_expr* *post_dec_expr*

index_expr

→ [→ *expr_list* →] →

call_expr

→ (→ *expr_list* →) →

compound_expr

→ *compound_stat* →

field_expr

→ . → **identifier** →

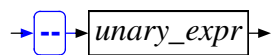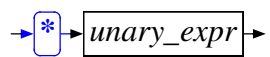*ptr_field_expr*

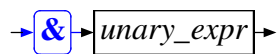→ -> → **identifier** →

post_inc_expr

→ ++ →

post_dec_expr

→ -- →

unary_expr
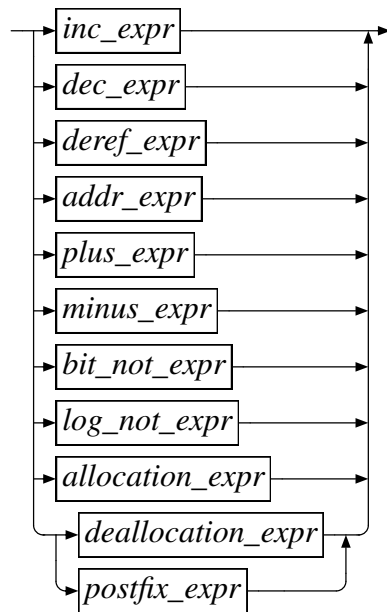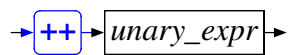
```
inc_expr
dec_expr
deref_expr
addr_expr
plus_expr
minus_expr
bit_not_expr
log_not_expr
allocation_expr
deallocation_expr
postfix_expr
```
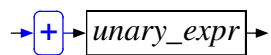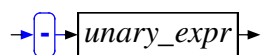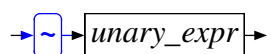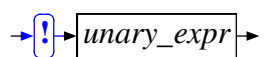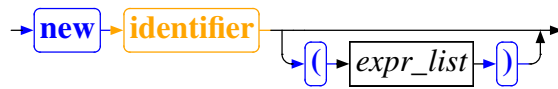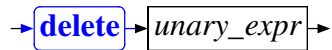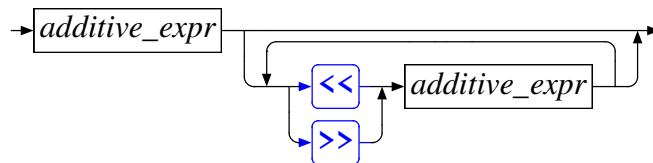
inc_expr

```
++ unary_expr
```

dec_expr

```
-- unary_expr
```

deref_expr

```
* unary_expr
```

addr_expr

```
& unary_expr
```

plus_expr

```
+ unary_expr
```

minus_expr

```
- unary_expr
```

bit_not_expr

```
~ unary_expr
```

log_not_expr

```
! unary_expr
```

allocation_expr

```
→ new → identifier ─────────────────────────────────→
→                      → ( → expr_list → ) ─↑
```

deallocation_expr

```
→ delete → unary_expr →
```

multiplicative_expr

```
→ unary_expr ─────────────────────────────→
              → ┌ * ┐ → unary_expr ─↑
                │ / │
                └ % ┘
```

additive_expr

```
→ multiplicative_expr ─────────────────────────→
                        → ┌ + ┐ → multiplicative_expr ─↑
                          └ - ┘
```

shift_expr

```
→ additive_expr ─────────────────────────────→
                 → ┌ << ┐ → additive_expr ─↑
                   └ >> ┘
```

relational_expr

```
→ shift_expr ─────────────────────────────→
              → ┌ <  ┐ → shift_expr ─↑
                │ >  │
                │ <= │
                └ >= ┘
```
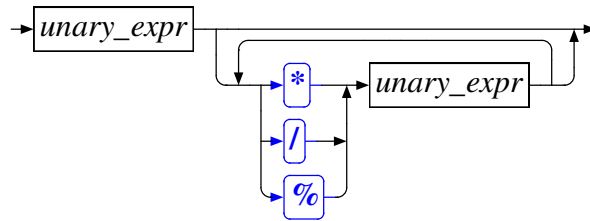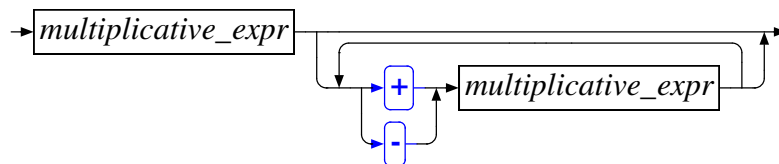
equality_expr
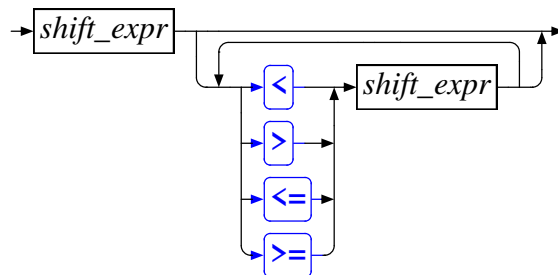
```
→ relational_expr ─────────────────────────────→
                   → ┌ == ┐ → relational_expr ─↑
                     └ != ┘
```
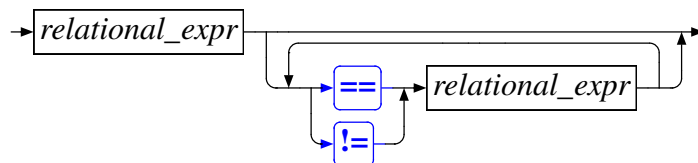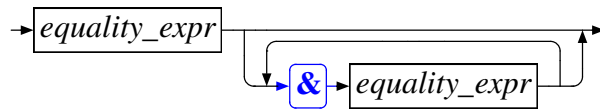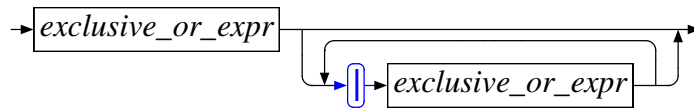
and_expr
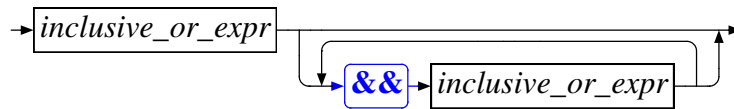

exclusive_or_expr
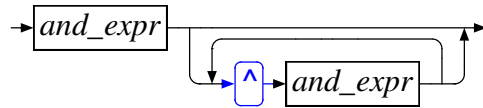
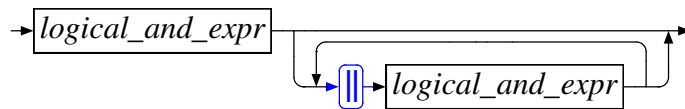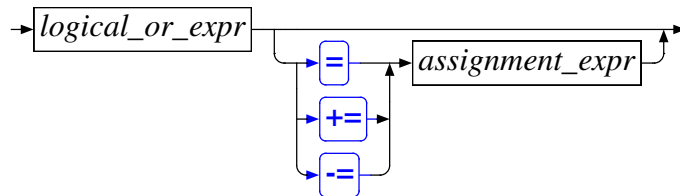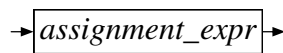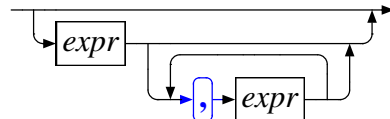*inclusive_or_expr*
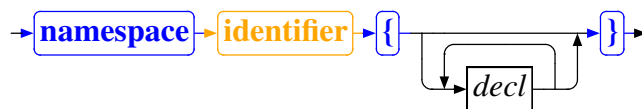

logical_and_expr

logical_or_expr

assignment_expr

expr

expr_list

namespace_decl

class_decl

→ **class** → **identifier** → **{** → *simple_decl* → **}** →

enum_decl

→ **enum** → **identifier** → **{** → *enum_item* → **,** → *enum_item* → **}** →

enum_item

→ **identifier** →

template_decl

→ **template** → **<** → *template_param* → **,** → *template_param* → **>** → *decl* →

template_param

→ **identifier** →

simple_decl

→ **identifier** → ***** → **identifier** → **=** → *expr* → **;** →
→ *compound_stat* →
→ *parameter_list* → *compound_stat* →
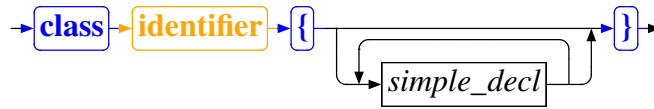
parameter_list

→ **(** → *parameter_decl* → **,** → *parameter_decl* → **)** →

parameter_decl

→ **identifier** → **identifier** →

empty_decl

→ **;** →

decl

```
    ┌─────────────────┐
───┬─▶│ namespace_decl  │──┬──▶
 ▶ │  └─────────────────┘  │
   ├─▶│ class_decl │───────▶│
▶▶ │  └────────────┘        │
   ├─▶│ enum_decl │────────▶│
▶▶ │  └───────────┘         │
   ├─▶│ template_decl │────▶│
▶▶ │  └───────────────┘     │
   ├─▶│ simple_decl │──────▶│
▶▶ │  └─────────────┘       │
   └─▶│ empty_decl │────────┘
 ▶  ▲▶└────────────┘
```

program

```
   ────────────────────────▶
      ┌──────────────┐
      │  ┌──────┐    │
      └─▶│ decl │────┘
         └──────┘
```