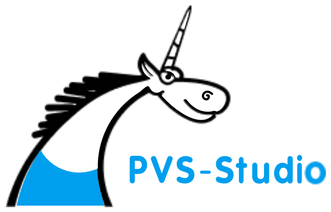
Parsing C++

And why it is tricky

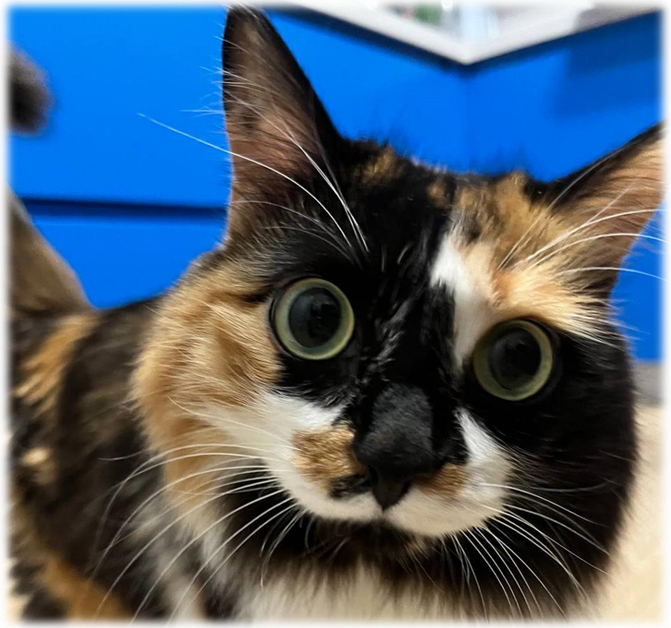


Yuri Minaev

Architect

Yuri Minaev

Architect at PVS-Studio



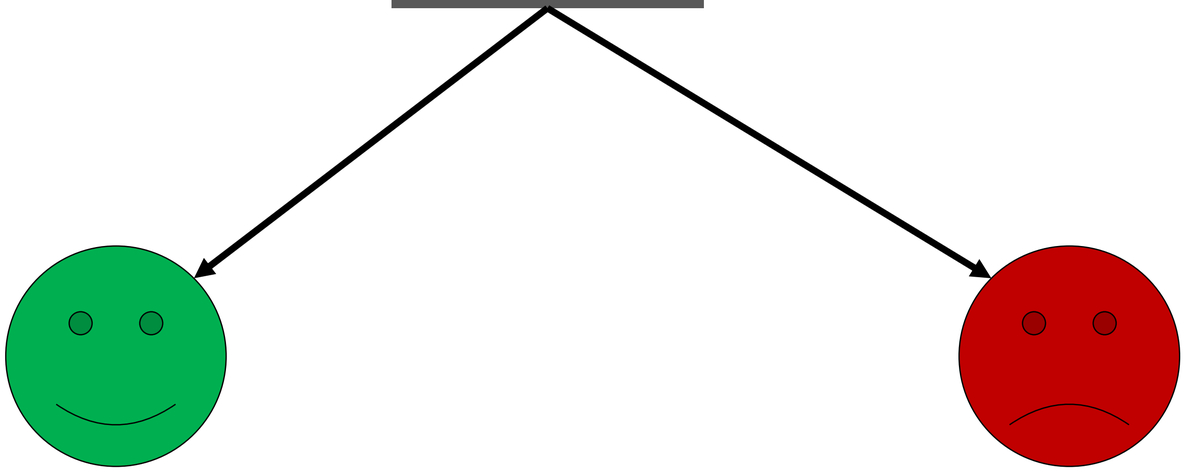
Parsing. What is it about?

```
minutes = left + seconds * 60
```

Lexer

```
<id> <=> <id> <+> <id> <*> <60>
```

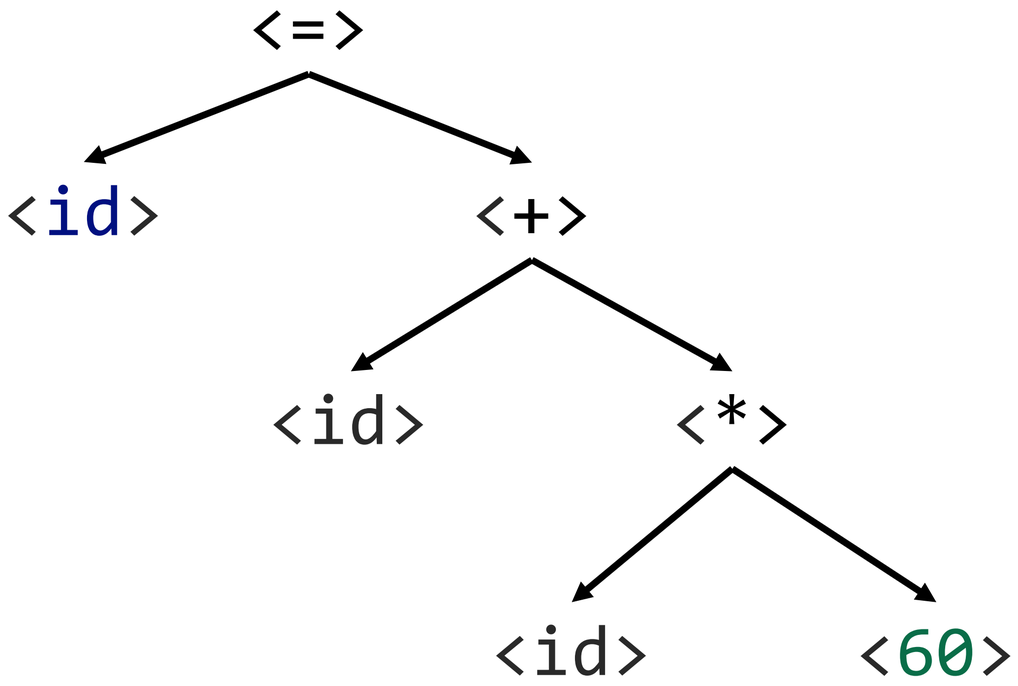
Parser



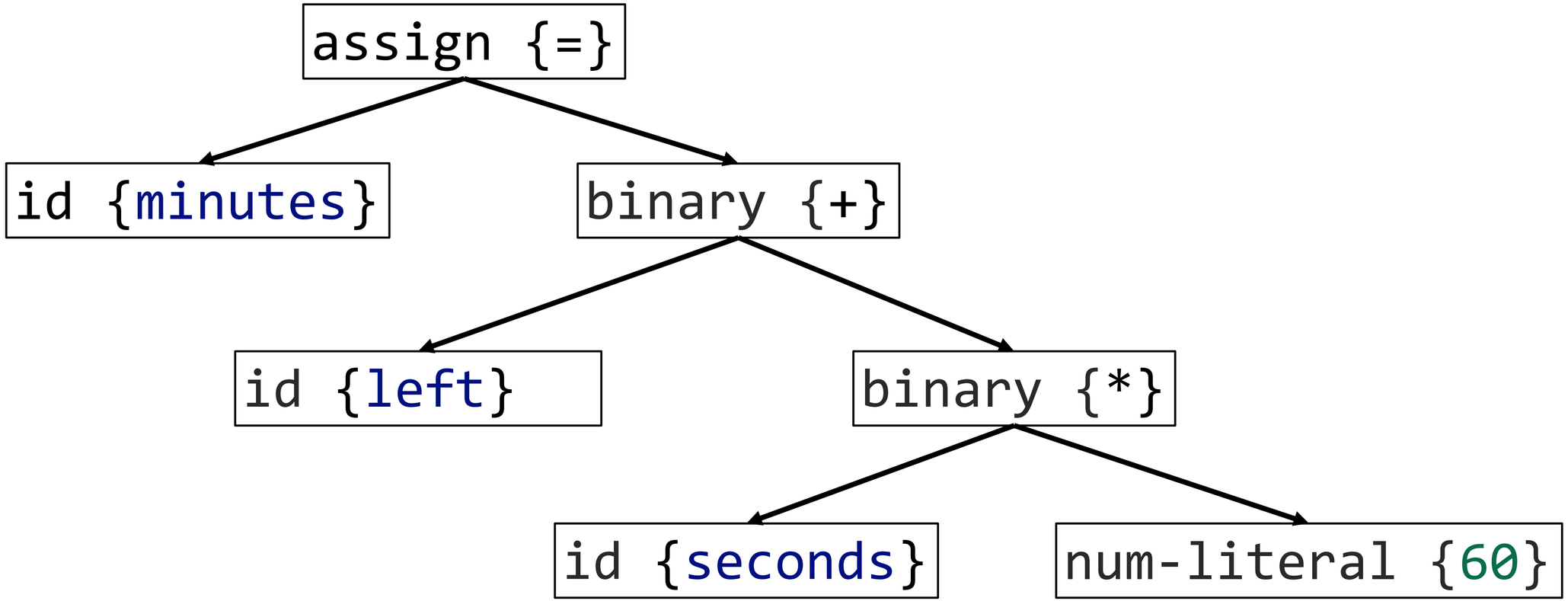
Parsing. What is it about?

```
<id> <=> <id> <+> <id> <*> <60>
```

Parsing. What is it about?



Parsing. The AST



Grammars

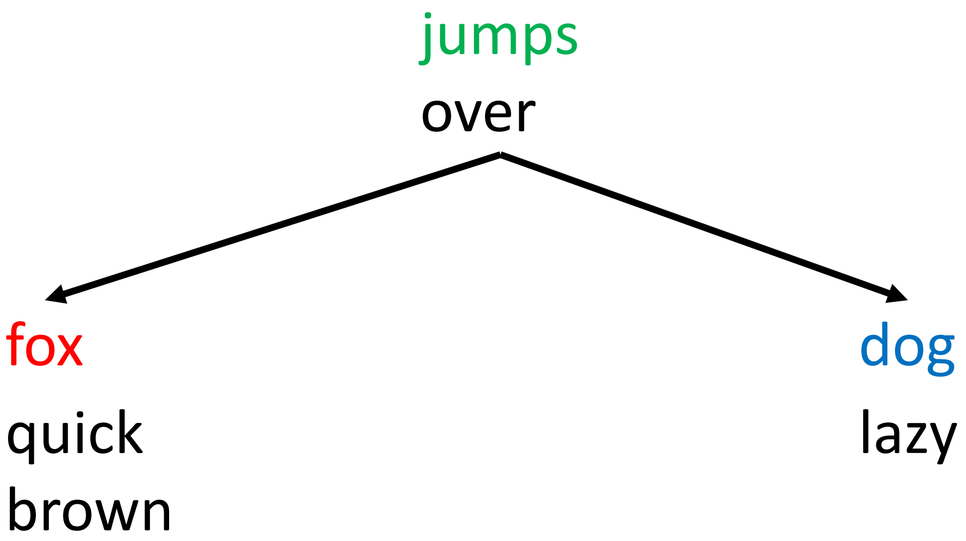
Grammars

The quick brown fox jumps over the lazy dog

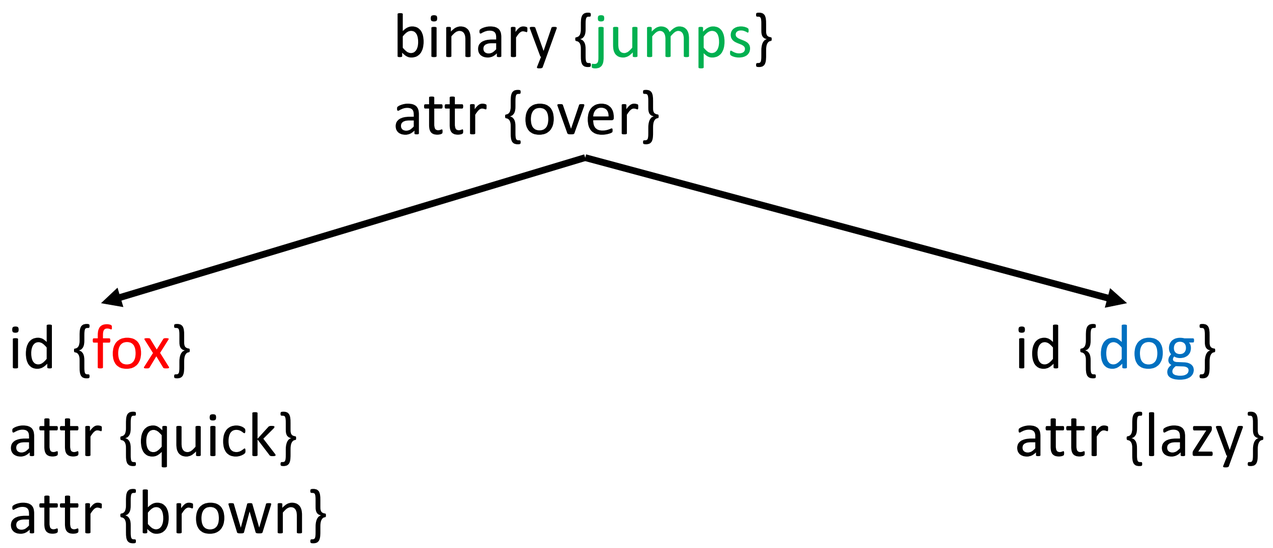
Grammars

The quick brown fox jumps over the lazy dog

Grammars



Grammars


binary {jumps}
attr {over}
id {fox}
attr {quick}
attr {brown}
id {dog}
attr {lazy}

Grammars

Regular

```
\#([a-fA-F]|[0-9]){3, 6}
```

Context-free

```
int a = 42;
```

Context-sensitive

```
Bach lang (aabccb, baabcaccb), ...
```

Ease of use for humans

Ease of parsing for machines

Grammars. Made-up language

expression-list:
    expression
    expression-list, expression

expression:
    binary-expr

binary-expr:
    id-expr
    binary-expr '$' id-expr

id-expr: one of
    a b c d

Grammars. Made-up language

expression-list:
  expression
  expression-list, expression

expression:
  binary-expr

binary-expr:
  id-expr
  binary-expr '$' id-expr

id-expr: one of
  a b c d

```
a
a, b, c
b $ c, a $ d
a $ b $ c $ d
```

Parsing

Parsing. Approaches

Descending

Ascending

a $ b $ c

Parsing. Descending

```
a $ b $ c
```

expression-list

Parsing. Descending

```
a $ b $ c
```

expression-list -> expression

Parsing. Descending

```
a $ b $ c
```

expression-list -> expression -> binary-expr

Parsing. Descending

```
a $ b $ c
```

expression-list -> expression -> binary-expr -> id-expr

Parsing. Descending

```
a $ b $ c
```

expression-list -> expression -> binary-expr -> id-expr -> a

Parsing. Descending

```
a $ b $ c
```

id-expr {a}

binary-expr

Parsing. Descending

```
a $ b $ c
```

id-expr {a}

binary-expr -> $

Parsing. Descending

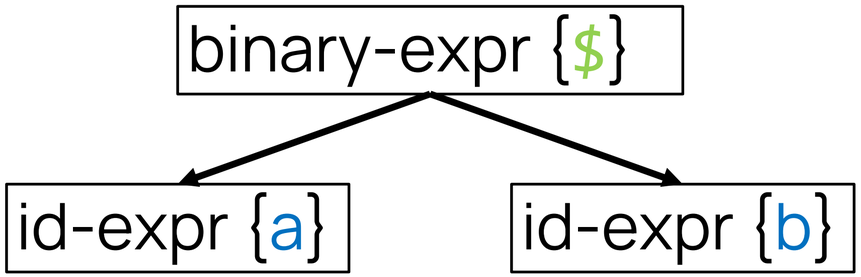
```
a $ b $ c
```

id-expr {a}

binary-expr -> $ -> id-expr -> b

Parsing. Descending

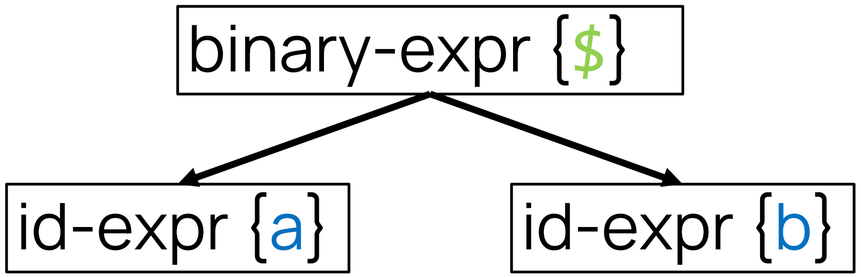
```
a $ b $ c
```



binary-expr

Parsing. Descending

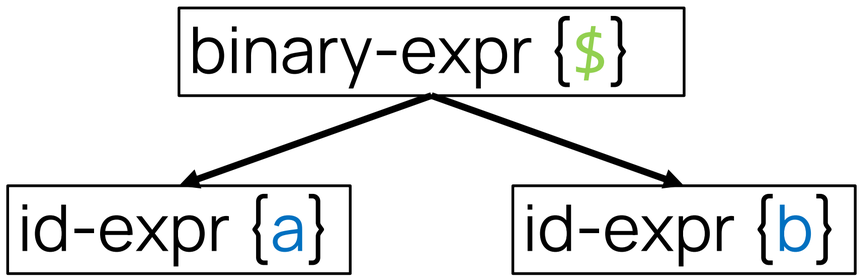
```
a $ b $ c
```



binary-expr -> $ -> id-expr -> c

Parsing. Descending

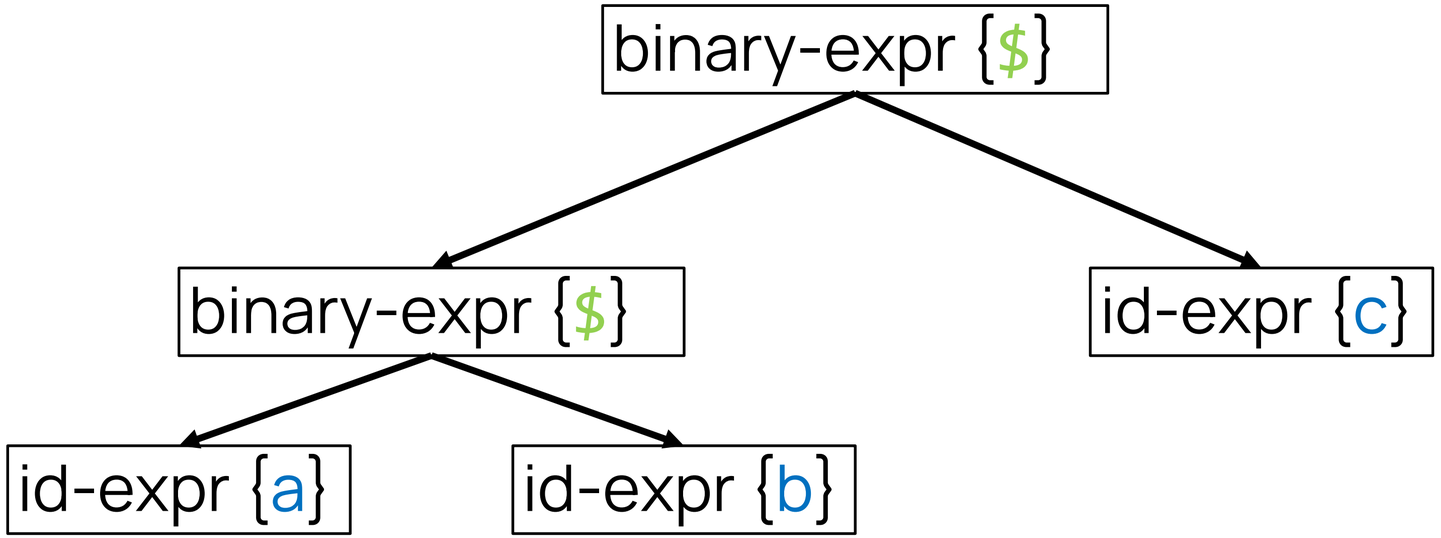
```
a $ b $ c
```



binary-expr -> $

Parsing. Descending

```
a $ b $ c
```



Parsing. Descending

```
// binary-expr:
//   id-expr
//   binary-expr '$' id-expr
```

Parsing. Descending

```
// binary-expr:
//   id-expr
//   binary-expr '$' id-expr
ast* binary_expr()
{
}
```

Parsing. Descending

```
// binary-expr:
//   id-expr
//   binary-expr '$' id-expr
ast* binary_expr()
{
  auto lhs = id_expr();
  return lhs;
}
```

Parsing. Descending

```
// binary-expr:
//   id-expr
//   binary-expr '$' id-expr
ast* binary_expr()
{
  auto lhs = id_expr();

  while(match('$'))
  {
    auto op  = consume();
    auto rhs = id_expr();
    lhs = make_binary(lhs, op, rhs);
  }

  return lhs;
}
```

Parsing. Descending

```
// binary-expr:
//   id-expr
//   binary-expr '$' id-expr
ast* binary_expr()
{
  auto lhs = id_expr();

  while(match('$'))
  {
    auto op  = consume();
    auto rhs = id_expr();
    lhs = make_binary(lhs, op, rhs);
  }

  return lhs;
}
```

Parsing. Descending

```
// binary-expr:
//   id-expr
//   binary-expr '$' id-expr ???
ast* binary_expr()
{
  auto lhs = id_expr();

  while(match('$'))
  {
    auto op  = consume();
    auto rhs = id_expr();
    lhs = make_binary(lhs, op, rhs);
  }

  return lhs;
}
```

Recursive descend tenet 1: Thou shalt not left-recurse

Parsing. Ascending

```
a $ b $ c
```

binary-expr:
 id-expr
 binary-expr '$' id-expr

id-expr: one of
 a b c d

Parsing. Ascending

```
$ b $ c
```

binary-expr:
 id-expr
 binary-expr '$' id-expr

id-expr: one of
 a b c d

a

Parsing. Ascending

```
$ b $ c
```

binary-expr:
 id-expr
 binary-expr '$' id-expr

id-expr: one of
 a b c d

id-expr {a}

Parsing. Ascending

```
$ b $ c
```

binary-expr:
 id-expr
 binary-expr '$' id-expr

id-expr: one of
 a b c d

binary-expr {a}

Parsing. Ascending

```
b $ c
```

binary-expr:
 id-expr
 binary-expr '$' id-expr

id-expr: one of
 a b c d

$
binary-expr {a}

Parsing. Ascending

```
$ c
```

binary-expr:
 id-expr
 binary-expr '$' id-expr

id-expr: one of
 a b c d

id-expr {b}
$
binary-expr {a}

Parsing. Ascending

```
$ c
```

binary-expr:
 id-expr
 binary-expr '$' id-expr

id-expr: one of
 a b c d

id-expr {b}
$
binary-expr {a}

Parsing. Ascending

```
$ c
```

binary-expr:
 id-expr
 binary-expr '$' id-expr

id-expr: one of
 a b c d

binary-expr {$}

Parsing. Ascending

c

binary-expr:
 id-expr
 binary-expr '$' id-expr

id-expr: one of
 a b c d

$
binary-expr {$}

Parsing. Ascending

binary-expr:
 id-expr
 binary-expr '$' id-expr

id-expr: one of
 a b c d

id-expr {c}
$
binary-expr {$}

Parsing. Ascending

```
binary-expr:
  id-expr
  binary-expr '$' id-expr

id-expr: one of
  a b c d
```

`binary-expr {$}`


GET ON
WITH IT!
memegenerator.net

C++ Grammar

Index	Summary	[gram]
A.1	General	[gram.general]
A.2	Keywords	[gram.key]
A.3	Lexical conventions	[gram.lex]
A.4	Basis	[gram.basis]
A.5	Expressions	[gram.expr]
A.6	Statements	[gram.stmt]
A.7	Declarations	[gam.dcl]
A.8	Modules	[gram.module]
A.9	Classes	[gram.class]
A.10	Overloading	[gram.over]
A.11	Templates	[gram.temp]
A.12	Exceptions handling	[gram.except]
A.13	Preprocessing directives	[gram.cpp]

Getting On With It

```
// binary-expr:
//   id-expr
//   binary-expr '$' id-expr
ast* binary_expr()
{
  auto lhs = id_expr();

  while(match('$'))
  {
    auto op  = consume();
    auto rhs = id_expr();
    lhs = make_binary(lhs, op, rhs);
  }

  return lhs;
}
```

Getting On With It

```
// additive-expression:
//   multiplicative-expression
//   additive-expression + multiplicative-expression
//   additive-expression - multiplicative-expression
ast::expr* add_expr()
{
  auto lhs = mul_expr();
  while(match('+', '-'))
  {
    auto op  = consume();
    auto rhs = mul_expr();
    lhs = ast::make_binary(lhs, op, rhs);
  }

  return lhs;
}
```

I HAZ EXPRESHUNZ

```
int meow()
{
  return 1 + []()
  {
  }();
}
```

I HAZ EXPRESHUNZ

```
int meow()
{
  return 1 + []()
  {
    struct oopsy
    {
    };
  }();
}
```

I HAZ EXPRESHUNZ

```
int meow()
{
  return 1 + []()
  {
    struct oopsy
    {
      int operator()()
      {
        return 42;
      }
    };
  }();
}
```

I HAZ EXPRESHUNZ

```
int meow()
{
  return 1 + []()
  {
    struct oopsy
    {
      int operator()()
      {
        return 42;
      }
    };
    oopsy o1;
    oopsy o2;
  }();
}
```

I HAZ EXPRESHUNZ

```
int meow()
{
  return 1 + []()
  {
    struct oopsy
    {
      int operator()()
      {
        return 42;
      }
    };
    oopsy o1;
    oopsy o2;
    return o1() + o2();
  }();
}
```

I HAZ EXPRESHUNZ

primary-expression:
 literal
 this
 (expression)
 id-expression
 lambda-expression
 fold-expression
 requires-expression

I HAZ EXPRESHUNZ

primary-expression:
 literal
 this
 (expression)
 id-expression
 lambda-expression
 fold-expression
 requires-expression

lambda-expression:
 lambda-introducer attribute-specifier-seq' lambda-declarator compound-statement
 lambda-introducer < template-parameter-list > requires-clause' attribute-specifier-seq'
 lambda-declarator compound-statement

I HAZ EXPRESHUNZ

primary-expression:
 literal
 this
 (expression)
 id-expression
 lambda-expression
 fold-expression
 requires-expression

lambda-expression:
 lambda-introducer attribute-specifier-seq' lambda-declarator **compound-statement**
 lambda-introducer < template-parameter-list > requires-clause' attribute-specifier-seq'
 lambda-declarator **compound-statement**

I HAZ EXPRESHUNZ

primary-expression:
 literal
 this
 (expression)
 id-expression
 lambda-expression
 fold-expression
 requires-expression

lambda-expression:
 lambda-introducer attribute-specifier-seq' **lambda-declarator compound-statement**
 lambda-introducer < template-parameter-list > requires-clause' attribute-specifier-seq'
 lambda-declarator compound-statement

I HAZ EXPRESHUNZ

statement:
 labeled-statement
 attribute-specifier-seq' expression-statement
 attribute-specifier-seq' compound-statement
 attribute-specifier-seq' selection-statement
 attribute-specifier-seq' iteration-statement
 attribute-specifier-seq' jump-statement
 attribute-specifier-seq' try-block
 declaration-statement

Declarations

```
int meow(int);
int meow(2);
```

Declarations

```
int meow(2);
```

Declarations

```
. int meow(2);
```

declaration:
 .name-declaration
  special-declaration

Declarations

```
. int meow(2);
```

name-declaration:
 .block-declaration
  nodeclspec-function-declaration
  function-definition
  deduction-guide
  empty-declaration

Declarations

```
. int meow(2);
```

block-declaration:
.simple-declaration ⬅ That's a lie

Declarations

```
. int meow(2);
```

simple-declaration:
 .decl-specifier-seq init-declarator-list' ;
 attribute-specifier-seq decl-specifier-seq init-declarator-list ;
 attribute-specifier-seq' decl-specifier-seq ref-qualifier' [identifier-list] initializer ;

Declarations

```
int . meow(2);
```

decl-specifier-seq -> ... -> simple-type-specifier -> **int**

Declarations

```
int . meow(2);
```

simple-declaration:
 decl-specifier-seq **.init-declarator-list'** ;

Declarations

```
int . meow(2);
```

init-declarator-list:
.init-declarator
init-declarator-list, init-declarator

Declarations

```
int . meow(2);
```

init-declarator:
.declarator initializer'

Declarations

```
int . meow(2);
```

declarator:
 ptr-declarator
 .noptr-declarator parameters-and-qualifiers

Declarations

```
int . meow(2);
```

noptr-declarator -> declarator-id -> ... -> id-expression ->
unqualified-id -> meow

Declarations

```
int meow . (2);
```

declarator:
 ptr-declarator
 noptr-declarator **.parameters-and-qualifiers**

Declarations

```
int meow . (2);
```

parameters-and-qualifiers:
.(**parameter-declaration-clause**) cv-qualifier-seq'
ref-qualifier' noexcept-specifier' attribute-specifier-seq'

Declarations

```
int meow (. 2);
```

parameters-and-qualifiers:
(**.parameter-declaration-clause**) cv-qualifier-seq'
ref-qualifier' noexcept-specifier' attribute-specifier-seq'

SyntaxError

Declarations

```
int meow(int);
int meow(2);
```

Declarations

```
int meow(int);
```

decl-specifier-seq declarator-id (parameter-declaration-clause);

```
int meow(2);
```

decl-specifier-seq declarator-id (expression-list);

Declarations

```
int meow(int);
```

decl-specifier-seq declarator-id (parameter-declaration-clause);

```
int meow(2);
```

decl-specifier-seq declarator-id (expression-list);

Recursive descend tenet 1: Thou shalt not left-recurse

Recursive descend tenet 1:
Thou shalt not left-recurse

Recursive descend tenet 2:
No prefix shall overlap with another

The Most Vexing Of Parses

```
struct paws{};
struct cat{ cat(...); };

void meow()
{
}
```

The Most Vexing Of Parses

```
struct paws{};
struct cat{ cat(...); };

void meow()
{
  cat kitty0 (paws);
  cat kitty1 (paws());
}
```

The Most Vexing Of Parses

```
struct paws{};
struct cat{ cat(...); };

void meow()
{
  cat kitty0 (paws);      // cat(paws)
  cat kitty1 (paws());    // cat(paws(void))
}
```

The Most Vexing Of Parses

```
struct paws{};
struct cat{ cat(...); };

void meow()
{
  cat kitty0 (paws);      // cat(paws)
  cat kitty1 (paws());    // cat(paws(void))
  cat kitty2 ((paws()));
}
```

The Most Vexing Of Parses

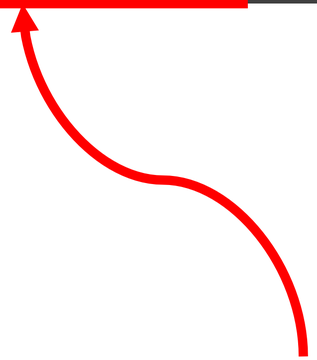
```
struct paws{};
struct cat{ cat(...); };

void meow()
{
  cat kitty0 (paws);      // cat(paws)
  cat kitty1 (paws());    // cat(paws(void))
  cat kitty2 ((paws())); // cat ctor called from temp paws
}
```

The Most Vexing Of Parses

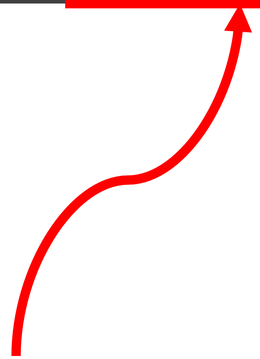
```
struct paws{};
struct cat{ cat(...); };

void meow()
{
  cat kitty0 (paws);      // cat(paws)
  cat kitty1 (paws());    // cat(paws(void))
  cat kitty2 ((paws())); // cat ctor called from temp paws

  cat kitty3 (paws(), paws(), paws(), paws());    // function
  cat kitty4 (paws(), paws(), paws(), (paws())); // variable
}
```

The Most Vexing Of Parses

```
cat kitty3 (paws(), paws(), paws(), paws());    // function
cat kitty4 (paws(), paws(), paws(), (paws())); // variable
```



Ambiguous



Disambiguation

Recursive descend tenet 1:
Thou shalt not left-recurse

Recursive descend tenet 2:
No prefix shall overlap with another

Recursive descend tenet 2:
No prefix shall overlap with another*

* Lest backtracking shall occur

Back We Go

```
int meow (. 2);
```

parameters-and-qualifiers:
(**.parameter-declaration-clause**) cv-qualifier-seq'
ref-qualifier' noexcept-specifier' attribute-specifier-seq'

Back We Go

```
int meow .(2);
```

Back We Go

```
int meow .(2);
```

init-declarator:
  declarator .initializer'

Back We Go

```
int meow .(2);
```

initializer:
  .brace-or-equal-initializer
  (expression-list)

Back We Go

```
int meow .(2);
```

initializer:
 brace-or-equal-initializer
 .(expression-list)

Back We Go

```
int meow (. 2);
```

initializer:
 brace-or-equal-initializer
 (**.expression-list**)

Back We Go

```
int meow (2 .);
```

initializer:
 brace-or-equal-initializer
 (expression-list .)

Back We Go

```
int meow (2) .;
```

initializer:
 brace-or-equal-initializer
 (expression-list).

Recursive descend tenet 2:
No prefix shall overlap with another*

* Lest backtracking should occur**

Recursive descend tenet 2:
No prefix shall overlap with another*

* Lest backtracking should occur**

** Backtacks lead to exponential complexity

Exponent

```
class no_mem_pool{};

template <size_t SIZE, class OTHER>
class mem_pool_list
{
public:
  mem_pool_list() {}
};
```

Exponent

```
typedef class
 mem_pool_list< 16,
 mem_pool_list< 32,
 mem_pool_list< 64,
 ...
 mem_pool_list< 512,
 mem_pool_list< 1024,
 mem_pool_list< 2048, no_mem_pool
 >>> ... >>> pools_t;
```

Exponent

```
typedef class
   mem_pool_list< 16,
      mem_pool_list< 32,
         mem_pool_list< 64,
            mem_pool_list< 512,
               mem_pool_list< 1024,
                  mem_pool_list< 2048, no_mem_pool >
                              >
                           >
                        >
                     >
                  > pools_t;
```

Exponent

```
typedef class
   mp< 16,
      mp< 32,
         mp< 64,
            mp< 512,
               mp< 1024,
                  mp< 2048, nmp >
                  >
               >
            >
         >
      > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent

```
typedef class
   mp< 16,
   ↑ mp< 32,
        mp< 64,
           mp< 512,
              mp< 1024,
                 mp< 2048, nmp >
                    >
                 >
              >
           >
        >
     > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent

```
typedef class
   mp< 16,
   ↑ mp< 32,
        mp< 64,
           mp< 512,
              mp< 1024,
                 mp< 2048, nmp >
                    >
                 >
              >
           >
        >
     > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent

```
typedef class
   mp< 16,
   ↑ mp< 32,
        mp< 64,
           mp< 512,
              mp< 1024,
                 mp< 2048, nmp >
                    >
                 >
              >
           >
        > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent

```
typedef class
   mp< 16,
     mp< 32,
     ↑ mp< 64,
         mp< 512,
           mp< 1024,
             mp< 2048, nmp >
               >
           >
         >
       >
     > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent

```
typedef class
   mp< 16,
      mp< 32,
         mp< 64,
            mp< 512,
               mp< 1024,
                  mp< 2048, nmp >
                  >
               >
            >
         >
      > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent

```
typedef class
   mp< 16,
      mp< 32,
         mp< 64,
            mp< 512,
               mp< 1024,
                  mp< 2048, nmp >
                  >
               >
            >
         >
      > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent

```
typedef class
   mp< 16,
      mp< 32,
         mp< 64,
            mp< 512,
               mp< 1024,
                  mp< 2048, nmp >
                  >
               >
            >
         >
      > pools_t;
```


simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent

```
typedef class
   mp< 16,
      mp< 32,
         mp< 64,
            mp< 512,
               mp< 1024,
                  mp< 2048, nmp >
                  >
               >
            >
         >
      > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent

```
typedef class
   mp< 16,
      mp< 32,
         mp< 64,
            mp< 512,
               mp< 1024,
                  mp< 2048, nmp >
                  >
               >
            >
         >
      > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' **simple-template-id**
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent

```
typedef class
   mp< 16,
      mp< 32,
         mp< 64,
            mp< 512,
               mp< 1024,
                  mp< 2048, nmp >
                  >
               >
            >
         >
      > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent

```
typedef class
   mp< 16,
      mp< 32,
         mp< 64,
            mp< 512,
               mp< 1024,
                  mp< 2048, nmp >
                  >
               >
            >
         >
      > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent

```
typedef class
   mp< 16,
      mp< 32,
         mp< 64,
            mp< 512,
               mp< 1024,
                  mp< 2048, nmp >
                  >
               >
            >
         >
      > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent

```
typedef class
   mp< 16,
      mp< 32,
         mp< 64,
            mp< 512,
               mp< 1024,
                  mp< 2048, nmp >
                  >
               >
            >
         >
      > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' **simple-template-id**
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Made-up language again

```
expression-list:
    expression
    expression-list, expression


expression:
    binary-expr
    joker-expr
```

```
joker-expr:
    binary-expr '$' 'X'
```

```
binary-expr:
    id-expr
    binary-expr '$' id-expr


id-expr: one of
    a b c d
```

Made-up language again

expression-list:
  expression
  expression-list, expression

expression:
  binary-expr
  joker-expr

joker-expr:
  binary-expr '$' 'X'

binary-expr:
  id-expr
  binary-expr '$' id-expr

id-expr: one of
  a b c d

```
a
a, b, c
b $ c, a $ d
a $ b $ c $ d
a $ X
a $ b $ X
```

Made-up language again

expression-list:
  expression
  expression-list, expression

expression:
  binary-expr
  joker-expr

joker-expr:
  binary-expr '$' 'X'

binary-expr:
  id-expr
  binary-expr '$' id-expr

id-expr: one of
  a b c d

```
a
a, b, c
b $ c, a $ d
a $ b $ c $ d
a $ X
a $ b $ X
```

Made-up language again

```
a $ b $ X
```

lexer_pos: 0

Made-up language again

```
a $ b $ X
```

lexer_pos: 1

expression-list -> expression -> binary-expr -> id-expr -> a

Made-up language again

a $ b $ X

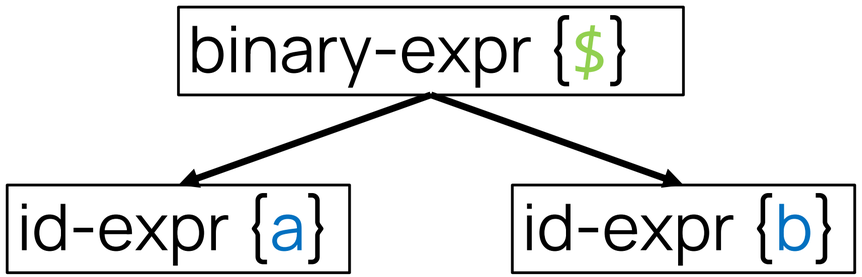
lexer_pos: 1

id-expr {a}

Made-up language again

a $ b $ X

lexer_pos: 3

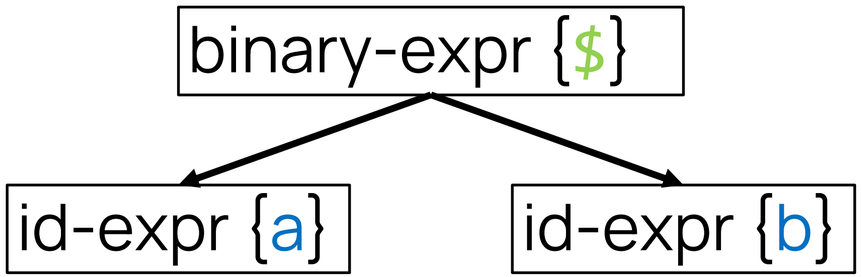


binary-expr -> $

Made-up language again

a $ b $ X

lexer_pos: 3

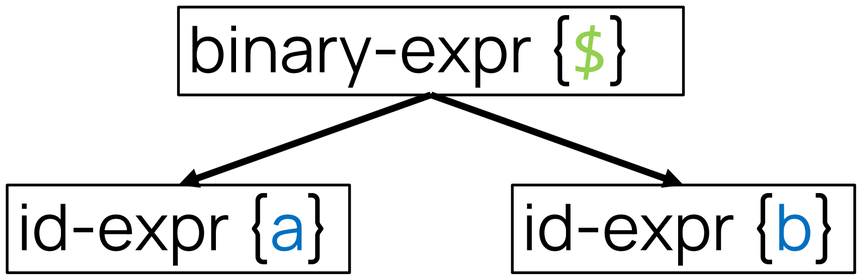


binary-expr{$}, 3

Made-up language again

a $ b $ X

lexer_pos: 4

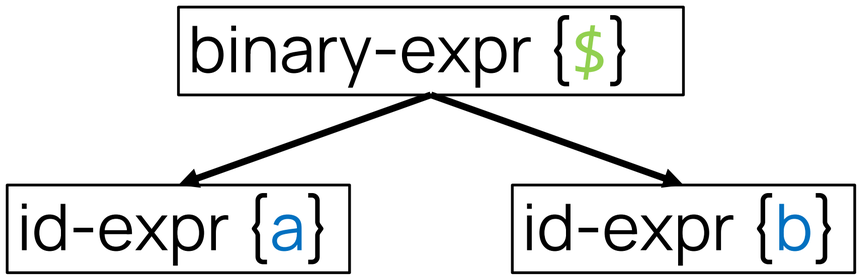


binary-expr{$}, 3

Made-up language again

a $ b $ X

lexer_pos: 4

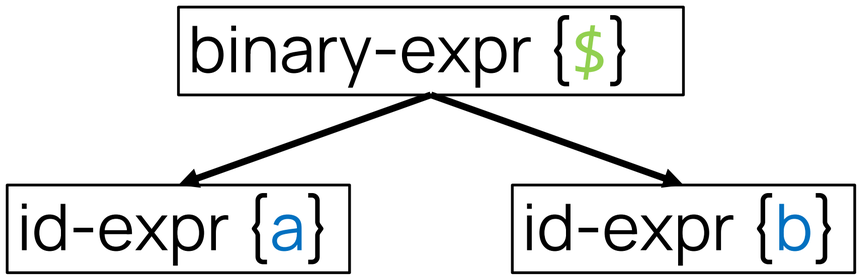


binary-expr{$}, 3

Made-up language again

a $ b $ X

lexer_pos: 0

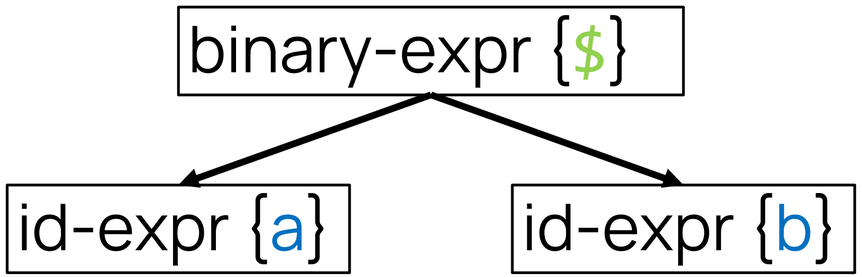


binary-expr{$}, 3

Made-up language again

a $ b $ X

lexer_pos: 0



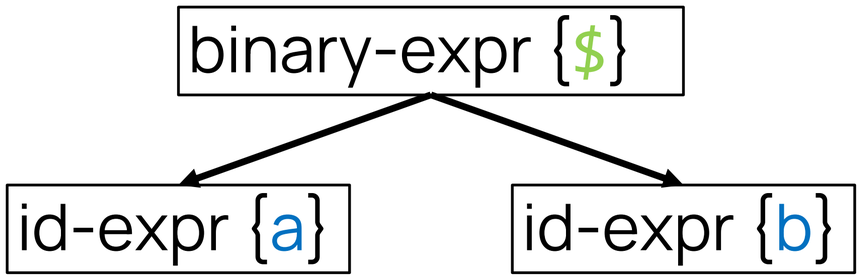
binary-expr{$}, 3

expression-list -> expression -> joker-expr ->binary-expr

Made-up language again

a $ b $ X

lexer_pos: 0



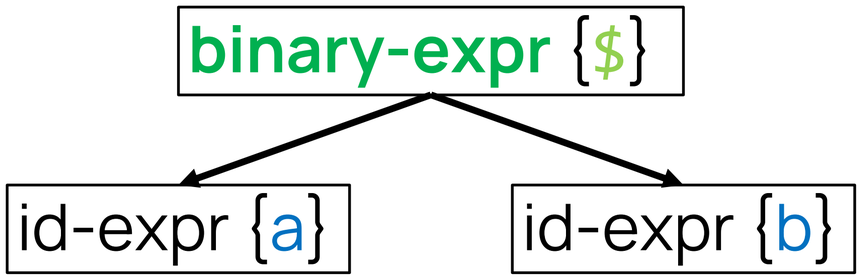
binary-expr{$}, 3

expression-list -> expression -> joker-expr -> **binary-expr**

Made-up language again

a $ b $ X

lexer_pos: 3

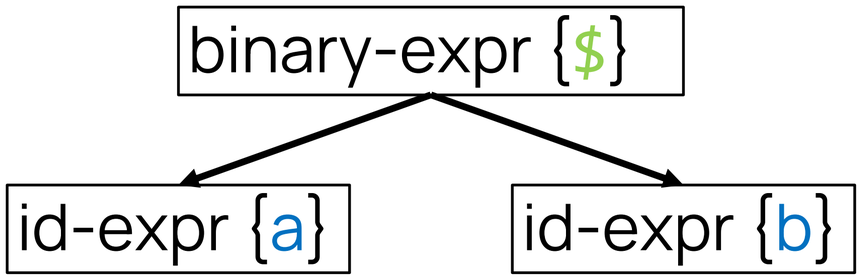


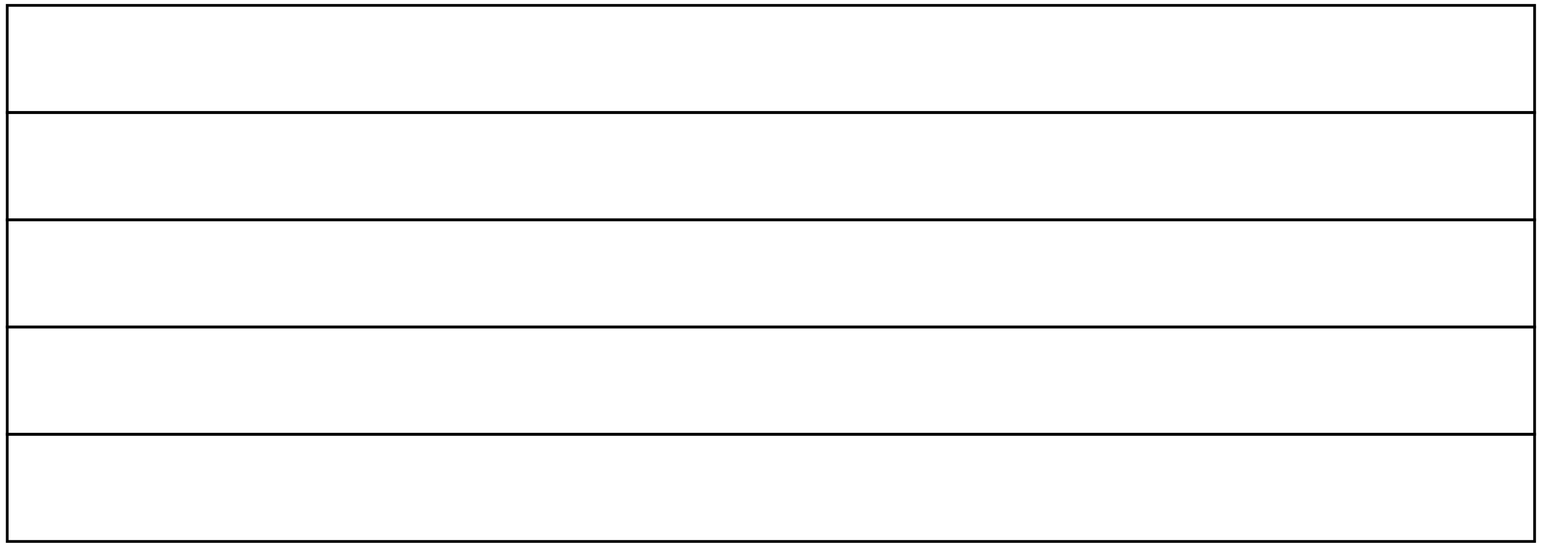
expression-list -> expression -> joker-expr -> **binary-expr**

Made-up language again

a $ b $ X

lexer_pos: 4



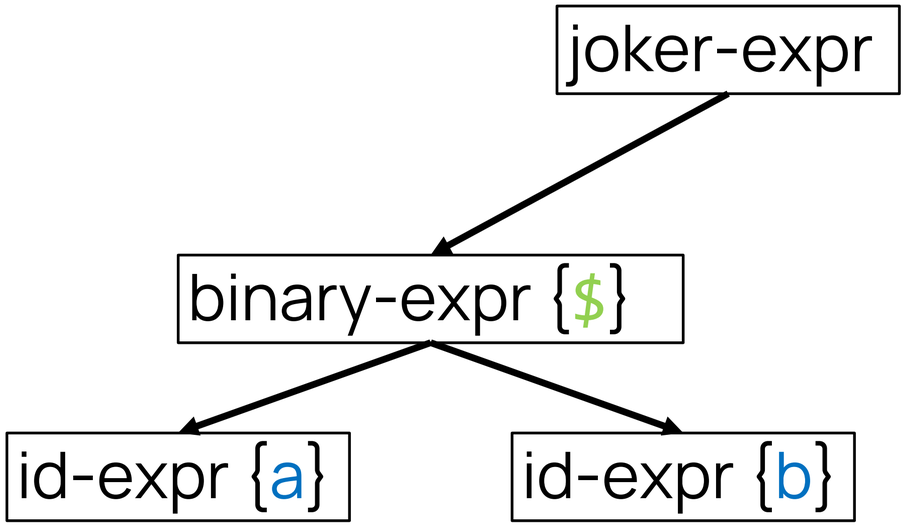


expression-list -> expression -> joker-expr

Made-up language again

```
a $ b $ X
```

lexer_pos: 5



Exponent Defeated

```
typedef class
   mp< 16,
   ↑ mp< 32,
        mp< 64,
           mp< 512,
              mp< 1024,
                 mp< 2048, nmp >
                    >
                 >
              >
           >
        >
     > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent Defeated

```
typedef class
   mp< 16,
      mp< 32,
         mp< 64,
            mp< 512,
               mp< 1024,
                  mp< 2048, nmp >
                  >
               >
            >
         >
      > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent Defeated

```
typedef class
   mp< 16,
      mp< 32,
         mp< 64,
            mp< 512,
               mp< 1024,
                  simple-template-id
                  >
               >
            >
         >
      > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent Defeated

```
typedef class
   mp< 16,
      mp< 32,
         mp< 64,
            mp< 512,
               simple-template-id
                  >
               >
            >
         > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent Defeated

```
typedef class
    mp< 16,
        simple-template-id
      > pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent Defeated

```
typedef class
    simple-template-id pools_t;
```

simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Exponent Defeated

```
typedef class
    simple-template-id pools_t;
```

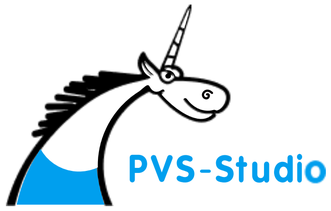
simple-type-specifier:
 nested-name-specifier' type-name
 nested-name-specifier' simple-template-id
 nested-name-specifier' template-name

nested-name-specifier:
 type-name ::
 namespace-name ::
 simple-template-id ::

Parsing C++

And why it is tricky

Q&A



Yuri Minaev

Architect