**Quiz 10**

1. **What are the type rules?**

* which types can be combined with certain operator
* assignment of expression to variable
* formal and actual parameters of a method call

1. **Give a code example of correct and type error use of variable?**

For example, in the code:

int x;

x := x + 1;

x.A := 1;

x[0] := 0;

the second line shows the correct type use of variable x: integer := integer + integer.

But the third and fourth line, it occurs the type errors use of variable x. The reasons are following.

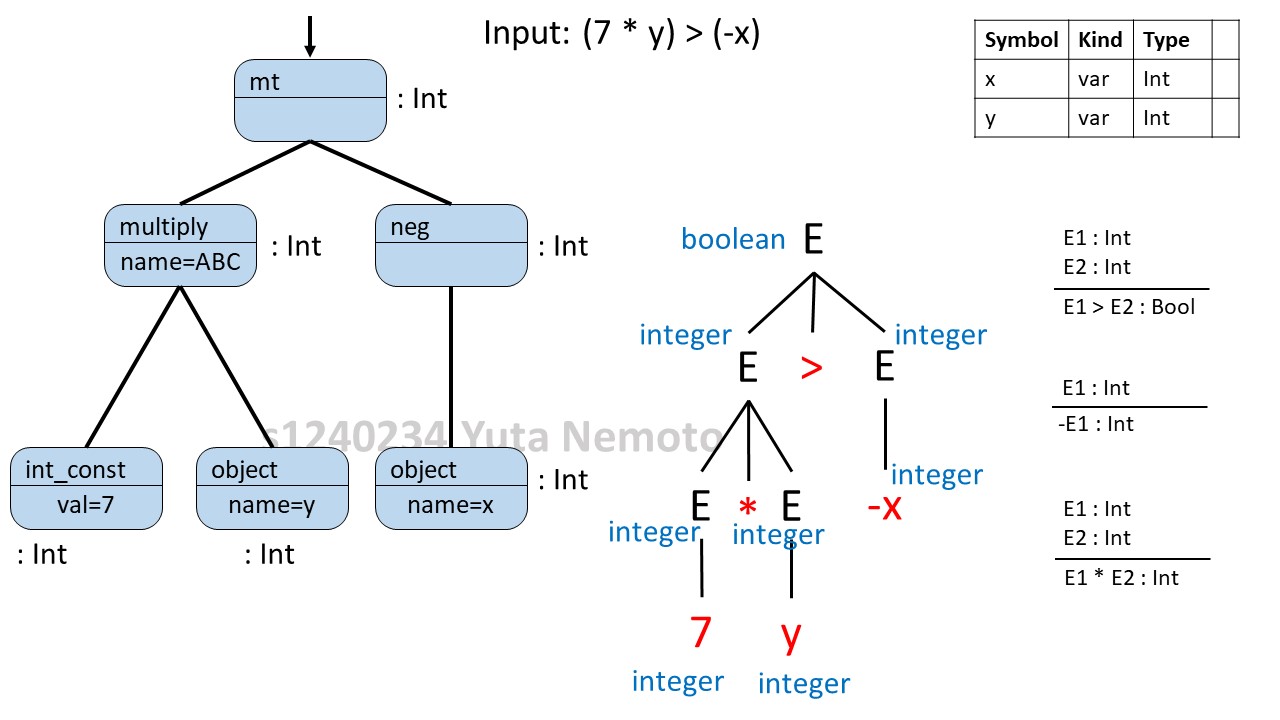
3rd line: Defines undefined element x.A, which type is not integer, as the integer value 1.

4th line: Although the variable x is declared as an integer variable, it tries to define x as an array.

1. **What are the main points for type checking implementation?**

* Single traversal over AST
* Types passed up the tree
* Type environment passed down the tree

1. **Give the ~~parse tree~~ AST, symbol table and type checking for the expression (7\*y) > (-x)?**



1. **Explain about static and dynamic types?**

**Dynamic type**: Class that is used in the new expression. It’s checked at a runtime, and even languages that are statically typed have dynamic types.

**Static type**: It captures all the dynamic types that the expression could have. It’s a compile-type notion.

1. **Write about type system?**

* First, a type is a set of values and associated operations.
* A type system is a collection of rules for assigning type expressions to various parts of the program
* Impose constraints that help enforce correctness.
* Provide a high-level interface for commonly used constructs (for example, arrays, records)
* Make it possible to tailor computations to the type, increasing efficiency (for example, integer vs. real arithmetic)

1. **Consider the language grammar given in table 1, write a valid program for this language, its parse tree with type assigned on the tree?**

Table 1: language grammar

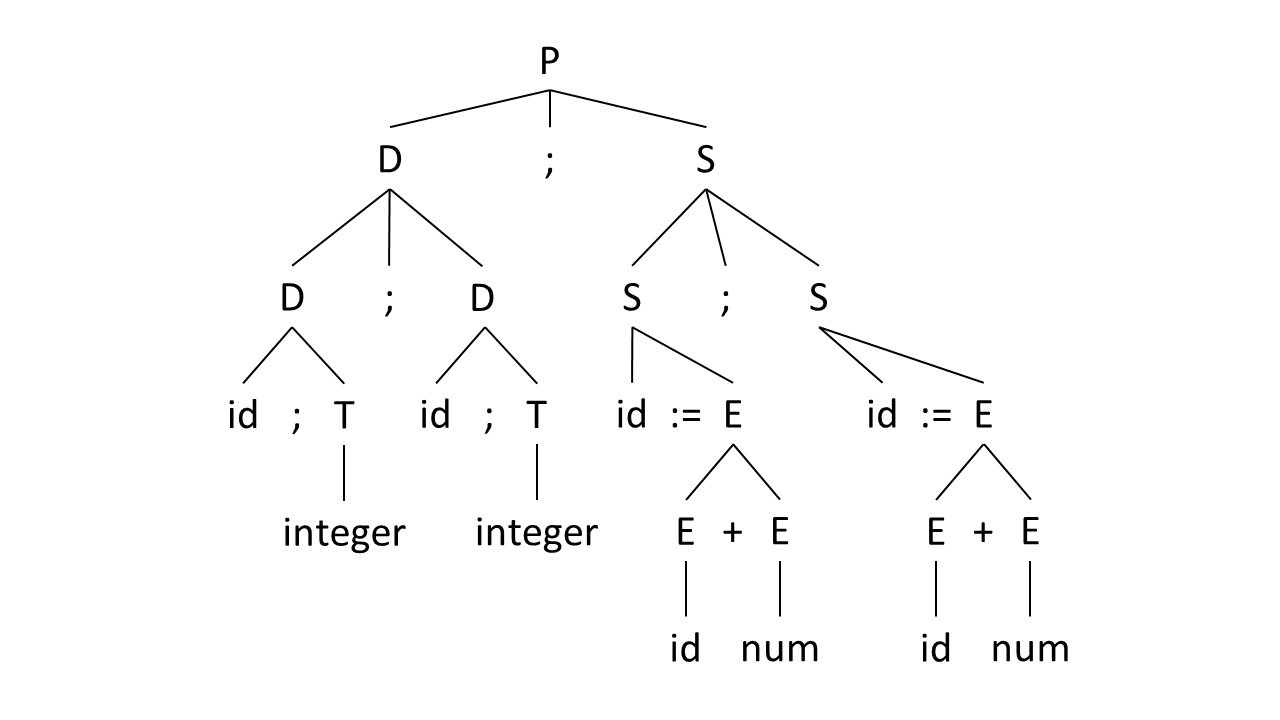
P → D ; S

D → D;D| id:T

T → integer | array [ num ] of T | ^T | T → T | T×T

S → S ; S | id := E | if E then S | while E do S

E → num | id | E + E | E [ E ] | E^ | E ( E )



i: integer; j: integer

i := i + 1;

j := i + 1;

Accumulate information about the declared type

D → D ; D

D → id : T

T → integer

T → array [ num ] of T1

T → ^T1

T → T1×T2

T → T1 → T2

{insert(id.name, T.type);}

{T.type = integer;}

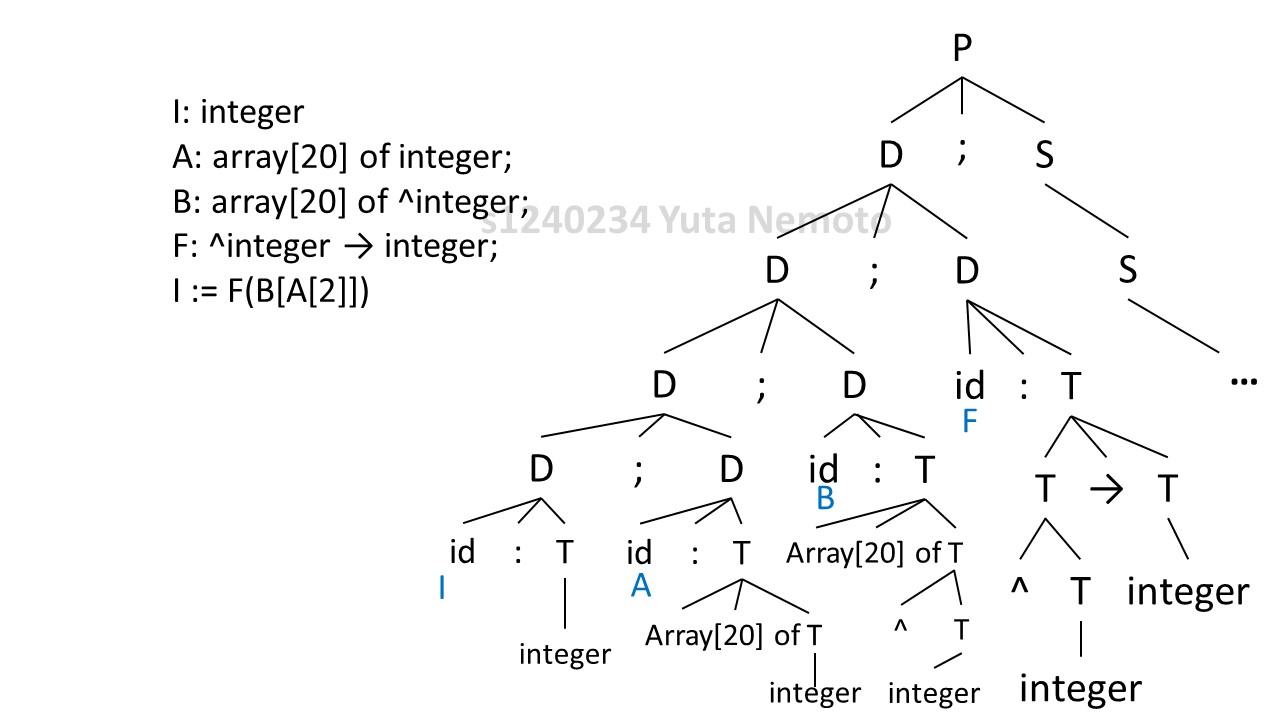
{T.type = array(T1.type,num);}

{T.type = pointer(T1.type);}

{T.type = product(T1.type,T2.type);}

{T.type = function(T1.type,T2.type);}

Parse Tree:



1. **Explain about components of type system and type equivalence?**

**Components of a Type System**

* Base Types: numbers, characters, Booleans.
* Compound/Constructed Types: array, string, enumerated types, record, pointer, classes (Object Oriented) and inheritance relationships, procedure/functions.
* Type Equivalence
* Inference Rules (Type checking)

etc.

**Type Equivalence**

There are two types: Structural and Name

For example,

Type A = Bool

Type B = Bool

If A and B match because they are both Boolean, then it’s the Structural type.

If A and B don’t match because they have different name, then it’s the Name type.