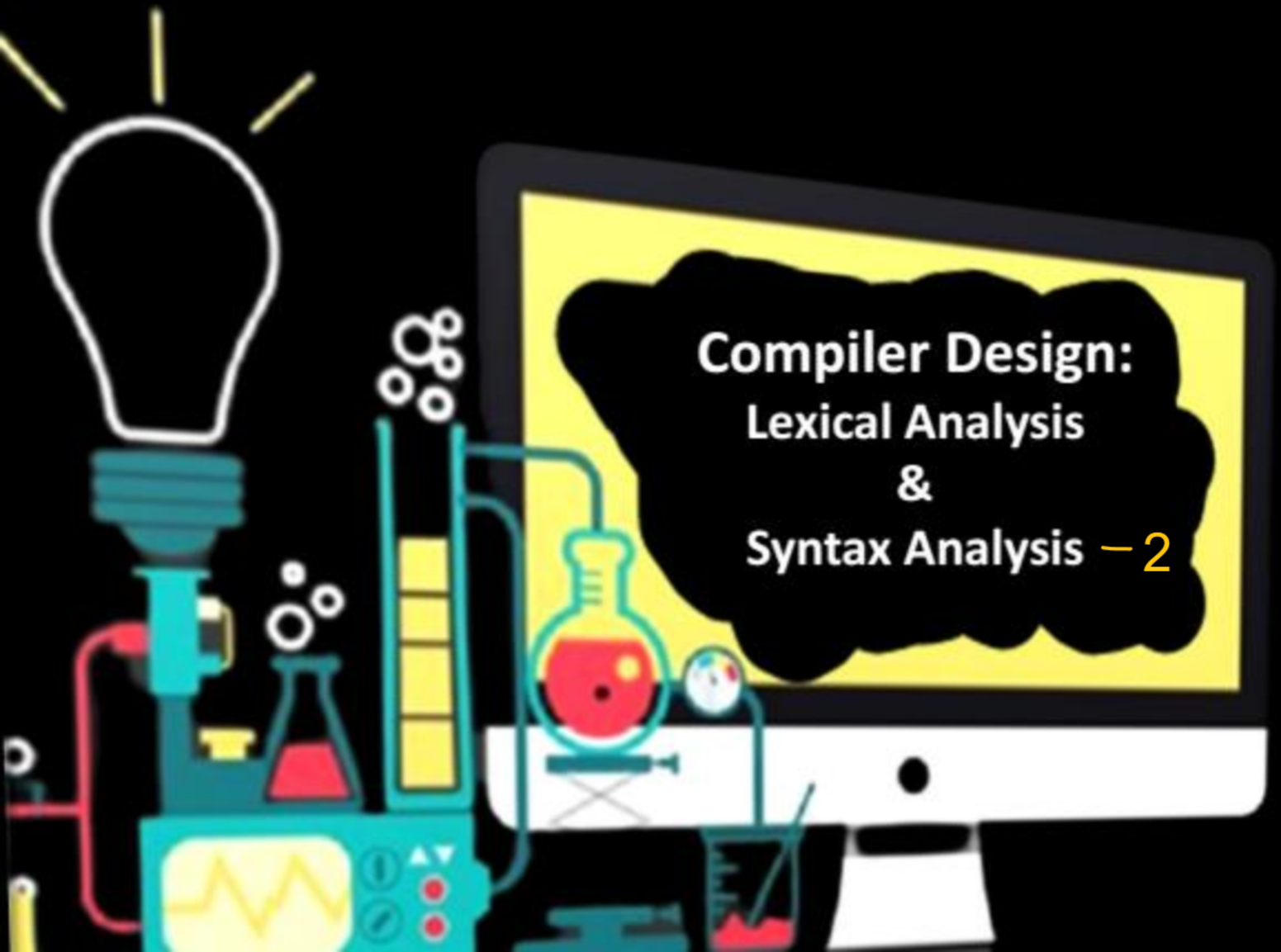
CS & IT Engineering



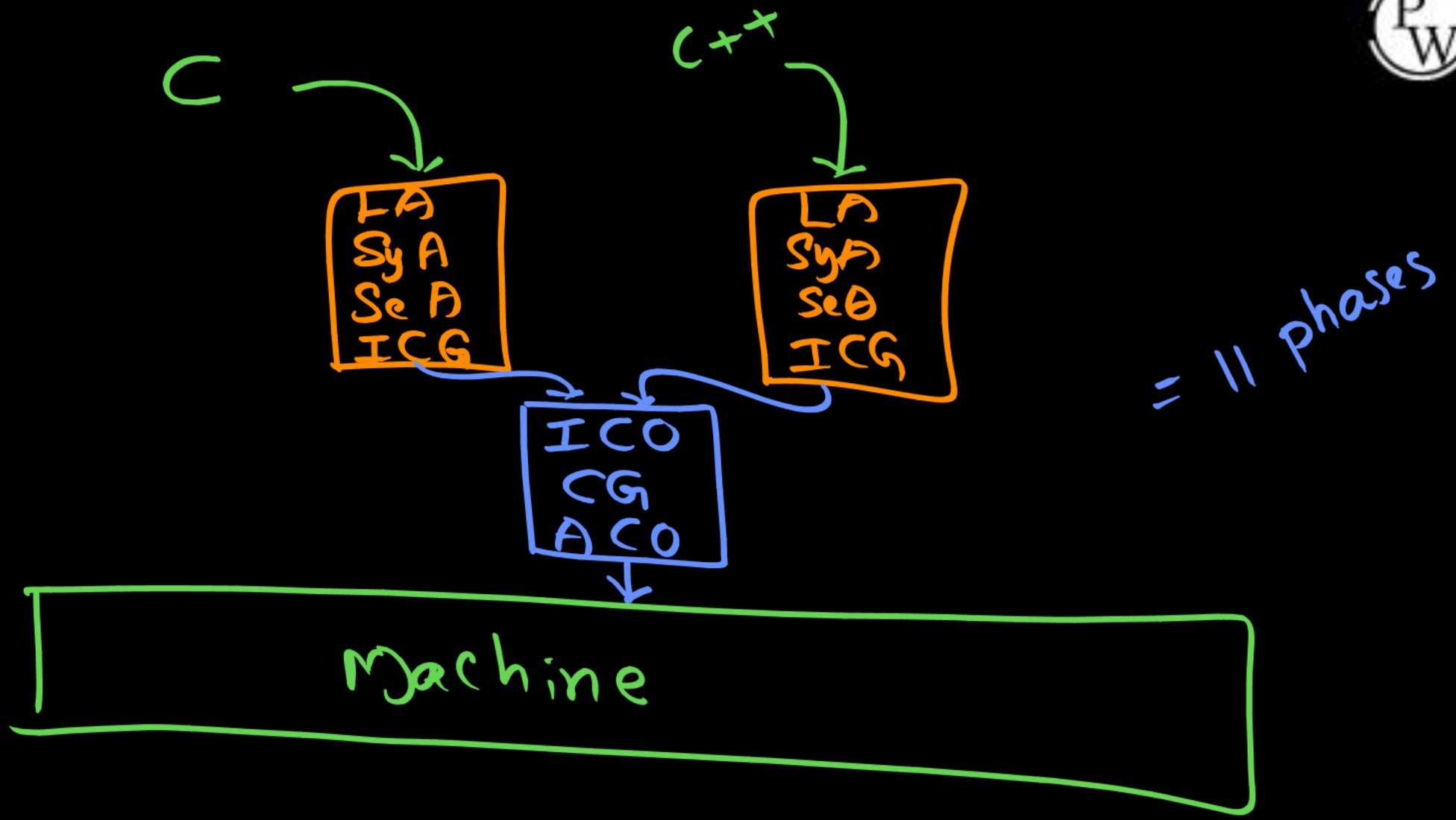
Compiler Design:
Lexical Analysis
&
Syntax Analysis – 2

Deva sir

Topics to be covered:

- → Lexical Analysis
- → Symbol Table
- → Errors

H.W:



= 11 phases

Compilation

(Static) Before Runtime

⇓

Begins from 1st character of file and only ends at the last character of file.

"File no need to have main() for compilation"

Execution

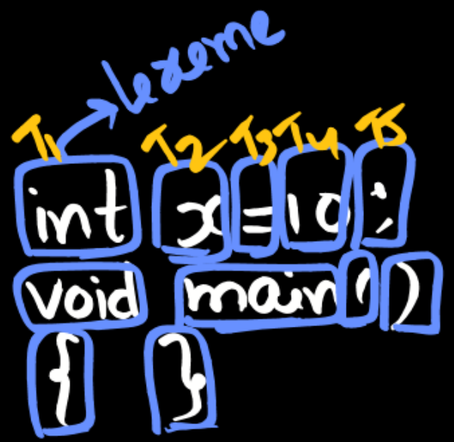
(Dynamic) During Runtime

⇓

Begins from main()

It is allowed only file that has main() to be executed.

Lexical Analysis



Source

Character stream

Sequence of characters

Lexical Analyzer

Token

Token stream
(Sequence of tokens)

Lexical Error

Lexical Analysis / Scanner / Linear phase / 1st phase / Token Recognizer / Token generator / Lexical phase

- It scans whole program character by character
- It groups characters into a "token" with the help of "Longest prefix Rule" (Maximal munch)
- It takes character stream as input and produces token stream as output.
- It may produce lexical errors if any

Lexical Analysis

- It recognizes tokens, white spaces, and comments
- It produces only tokens.
* It may take help of Parser to separate tokens
* It also uses/creates Symbol table

Symbol Table

lexeme	token	Line	Type	...
fun	id_1	1		
x	id_2	2	int	
y	id_3	3	char	

```
void fun( ) {
    int x = 10;
    char y;
}
```

Literal Table

10

- It is data structure
- It stores attribute information of each identifier
- Depends on scope, compiler may create one or more Symbol table.

Lexical Analysis

1. Lexeme : It is smallest unit of program (Logic)
2. Token : It represents lexeme
3. Types of Tokens
4. Finding Tokens
5. Finding Lexical Errors

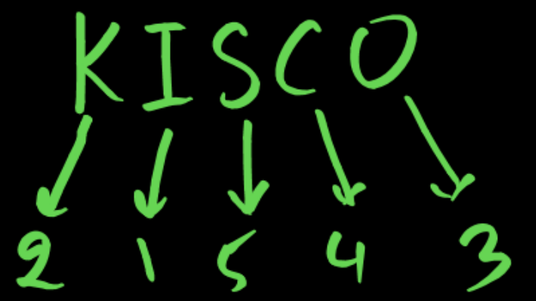
Lexical Analysis

```
int  x ;
```

Lexeme	TOKen (Internal representation of lexeme)
int	T_1
x	T_2
;	T_3

Types of TOKens :

1) Identifiers
2) Keywords
3) Operators
4) Literals / Constants
5) Special



Lexical Analysis

1) Identifier → Name of variable / function

↳ Rule : i) contains only letters, digits, and underscore
ii) Should not start with digit.
iii) Should not be a Keyword

Find Identifiers:

x ✓

x123 ✓

1x23 ✗

int ✗

Int ✓

int1 ✓

2) Keywords : → 32 keywords

int	auto	signed	const	struct	if	sizeof
float	static	unsigned	volatile	union	else	
char	register			enum	switch	
double	extern			typedef	case	
short					default	
long					break	
void					while	
					do	
					for	
					continue	
					return	
					goto	

3) Operators:

i) Arithmetic +, -, *, /, %

ii) Relational <, <=, >, >=

iii) Logical &&, ||, !

iv) Bitwise &, |, ~, ^, <<, >>

v) Unary +, -, ++, --, &, *, (type), !, ~, sizeof

vi) Assignment =, +=, -=, /=, *=, <<=, >>=, ...

vii) Special operators: →, ., ,

( ), [ ], ?:

4) Literals / constants :

Integer
- Decimal (0 to 9) : 2391
- Hexa (0 to F) : 0x2ad3, 0X2AD3
- Octal (0 to 7) : 0237

character : 'a' '\n' 'A'

Real (float/double) : 23.64f, 23.64, 10e-2

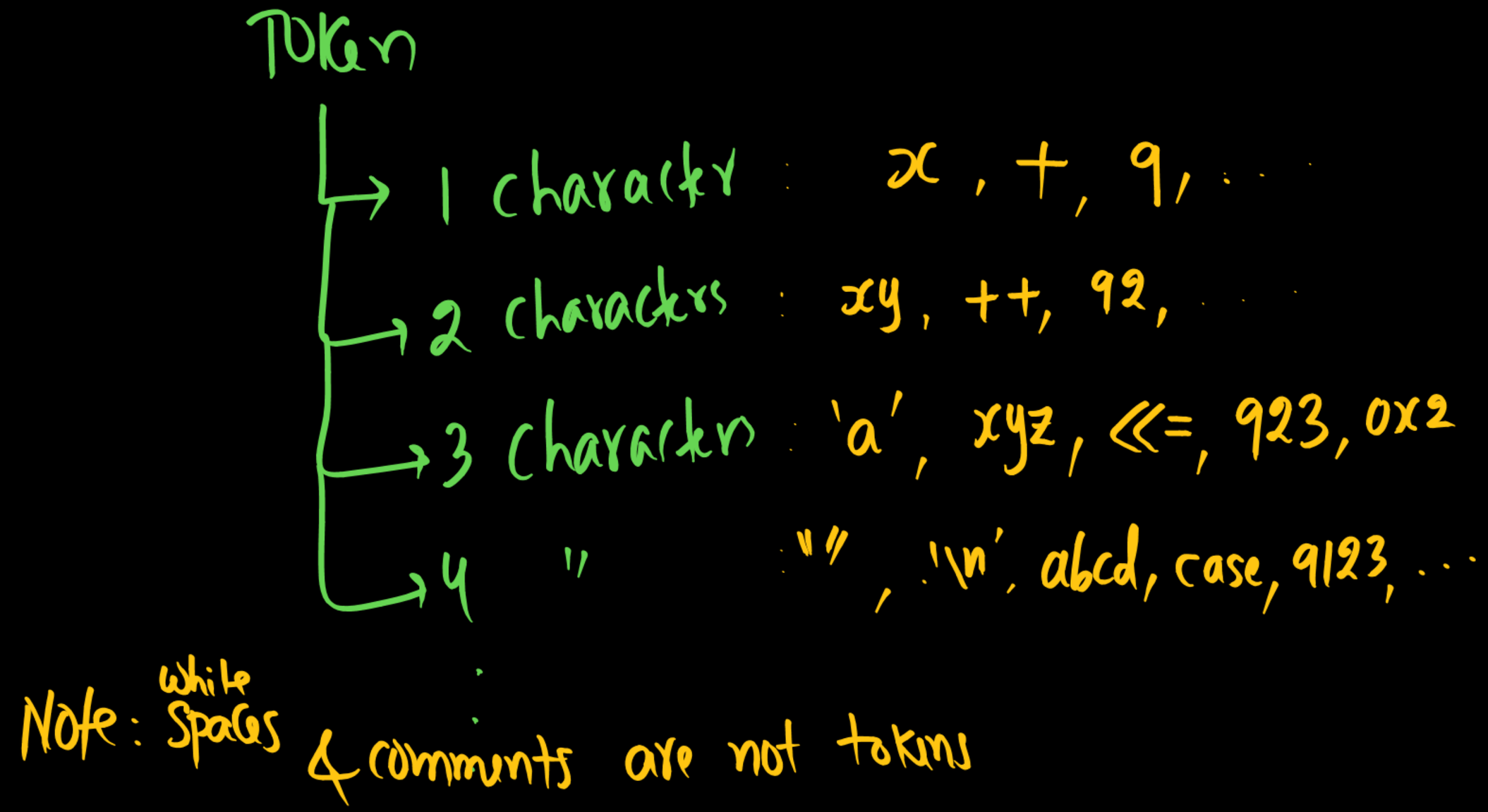
String : "gate"

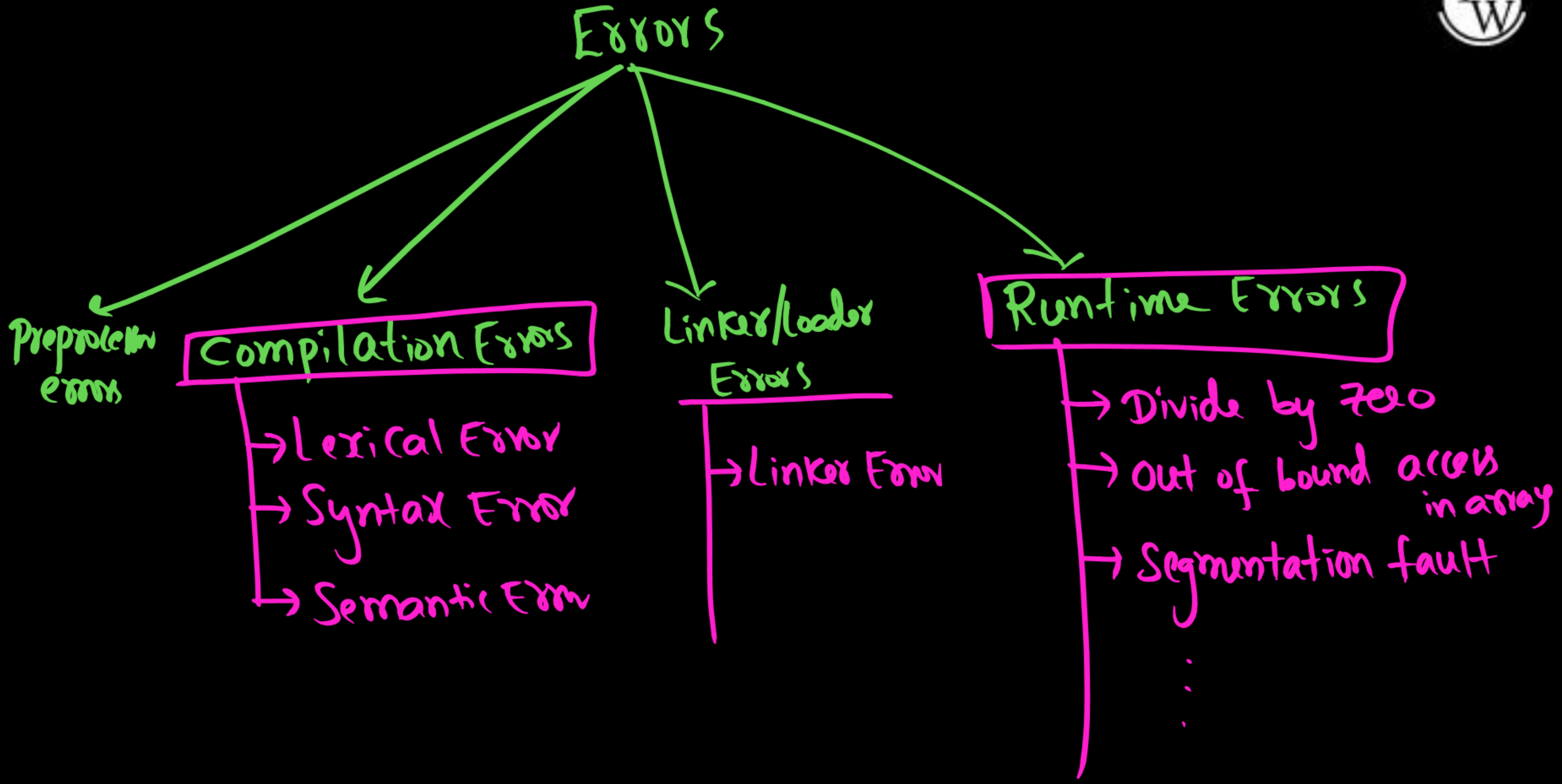
1 token
1 character constant

5) Special tokens:

[ ] ( ) { } ;

Lexical Analysis




Errors
Preprocessor errors
Compilation Errors
Lexical Error
Syntax Error
Semantic Error
Linker/loader Errors
Linker Error
Runtime Errors
Divide by zero
Out of bound access in array
Segmentation fault

Find the tokens.

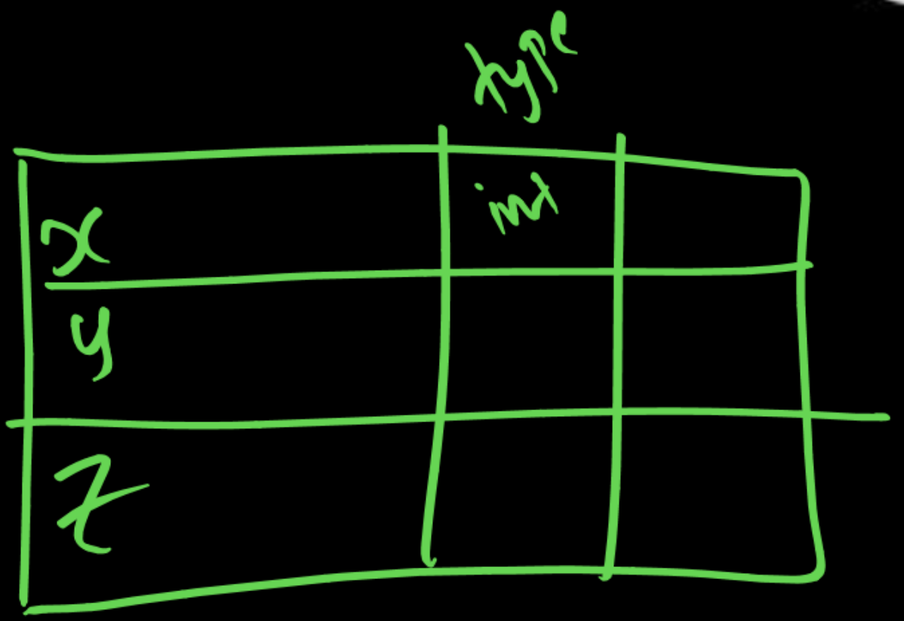
① `int x = 10 ;` = 5 — No Error

② `int x , y ;` = 5 — No error

③ `int x = y = z ;` = 7 — Semantic Error: y & z are not declared

④ `int float , char ;` = 5 — Syntax Error / compilation Error

```
int x;
y=z;
x=y;
```



Lexical Analysis

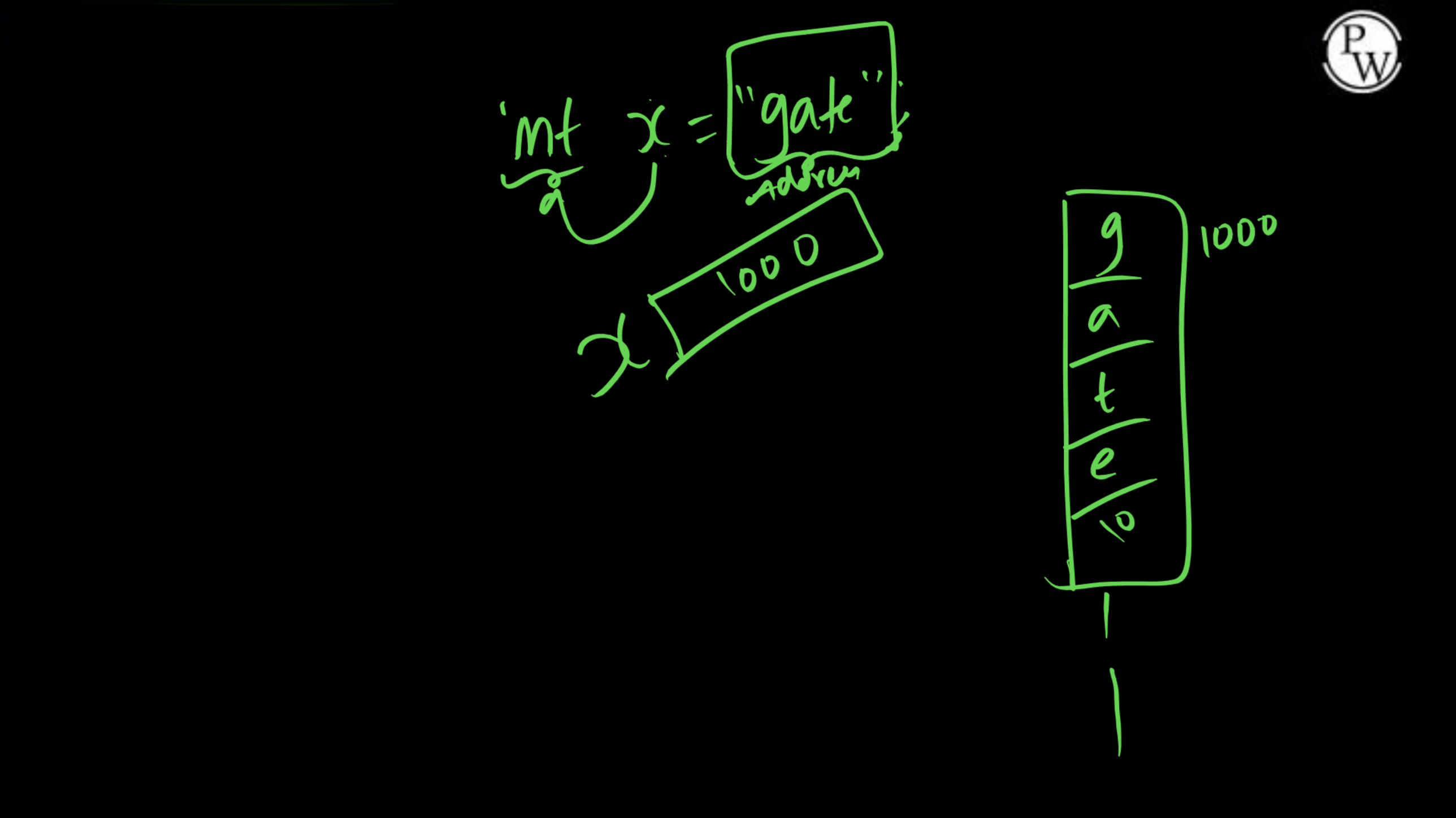
⑤ `x = "gate" ;` = 4 tokens, No lexical error, No syntax error, Semantic error

⑥ `int x = "gate" ;` = 5 tokens, No Lexical error, No syntax error, Semantic error

⑦ `int *p ;` = 4 tokens, No error

⑧ `int *p = 1000 ;` = 6 tokens, Semantic error

⑨ `int *p = (int *) 1000 ;` = 10 tokens, No error, p = 1000 (address), integer


int x = "gate";
address
x
1000
g
a
t
e
\0
1000

Lexical Analysis

(10)
```
void main( )
{
   printf("x = %d", x);
}
```

(11) `x = a+++++++==**b;`

(12) `int **p;`

(13) `x++;`

Find no. of tokens.

Thank you