Lexical Analysis

Rakib Mahmud

Grammar

G is a grammar, which consists of a set of production rules. It is used to generate the strings of a language.

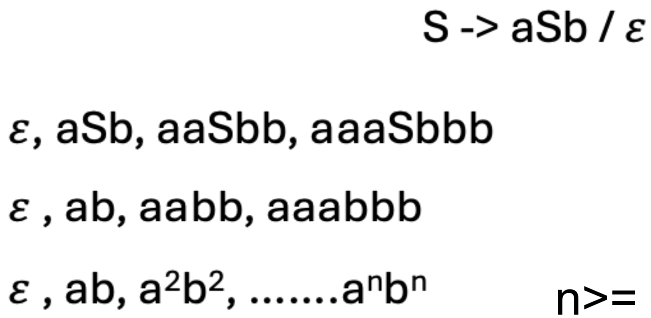
G= { V, T, P, S }

V= Variable (Represented with capital letter)

T= Terminal (Represented with small letter)

P= Production Rule

S= Start symbol



S -> aSb / ε

ε, aSb, aaSbb, aaaSbbb

ε , ab, aabb, aaabbb

ε , ab, a^2b^2, ...a^nb^n

n>= 0

DFA

- In DFA, for each input symbol, one can determine the state to which the machine will move. Hence, it is called Deterministic Automaton.
- As it has a finite number of states, the machine is called Deterministic Finite Machine or Deterministic Finite Automaton.

Formal Definition of a DFA

- A DFA can be represented by a 5-tuple (Q, ∑, δ, q_0, F) where −
  - Q is a finite set of states.
  - ∑ is a finite set of symbols called the alphabet.
  - δ is the transition function where $\delta: Q \times \sum \rightarrow Q$
  - q_0 is the initial state from where any input is processed ($q_0 \in Q$).
  - F is a set of final state/states of Q ($F \subseteq Q$).

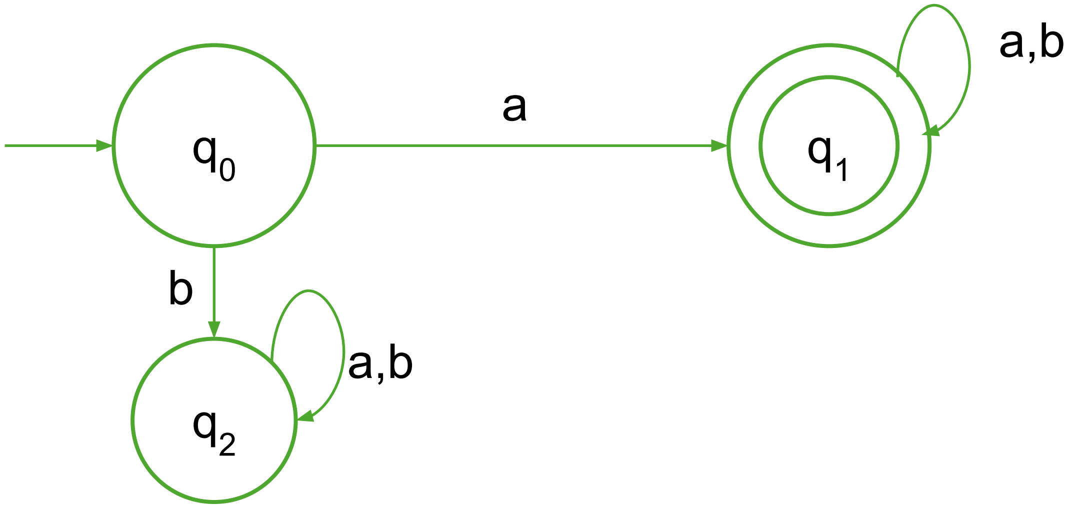
Graphical Representation of a DFA

- A DFA is represented by digraphs called state diagram.
- The vertices represent the states.
- The arcs labeled with an input alphabet show the transitions.
- The initial state is denoted by an empty single incoming arc.
- The final state is indicated by double circles.

Example

- **Make a DFA where all strings start with a.**
- You have two alphabets (a,b)
  - { a, aa, aaa, ab, ba }

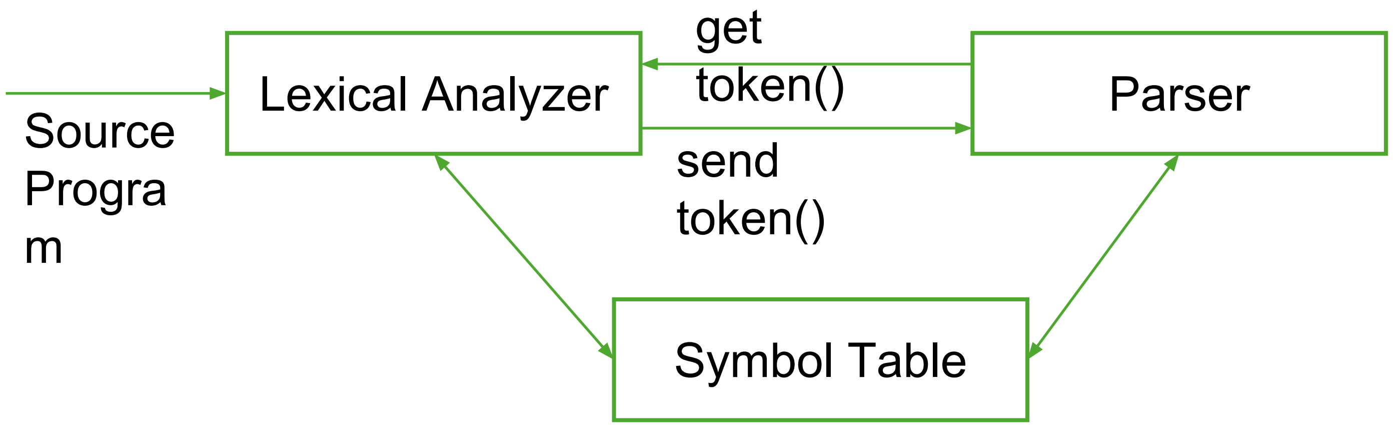
✗



Lexical Analyzer

- Lexical analyzer divides the given program into meaningful words which is known as tokens.
- Tokens are normally identifiers, separator, keywords, operators, constant and special symbol.
- The program which did this is called lexer, tokenizer, scanner.
- Lexical analyzer eliminates comments, white space character. (blank, Tab)
- It helps in giving error message such as exceeding length by providing row no and column no.

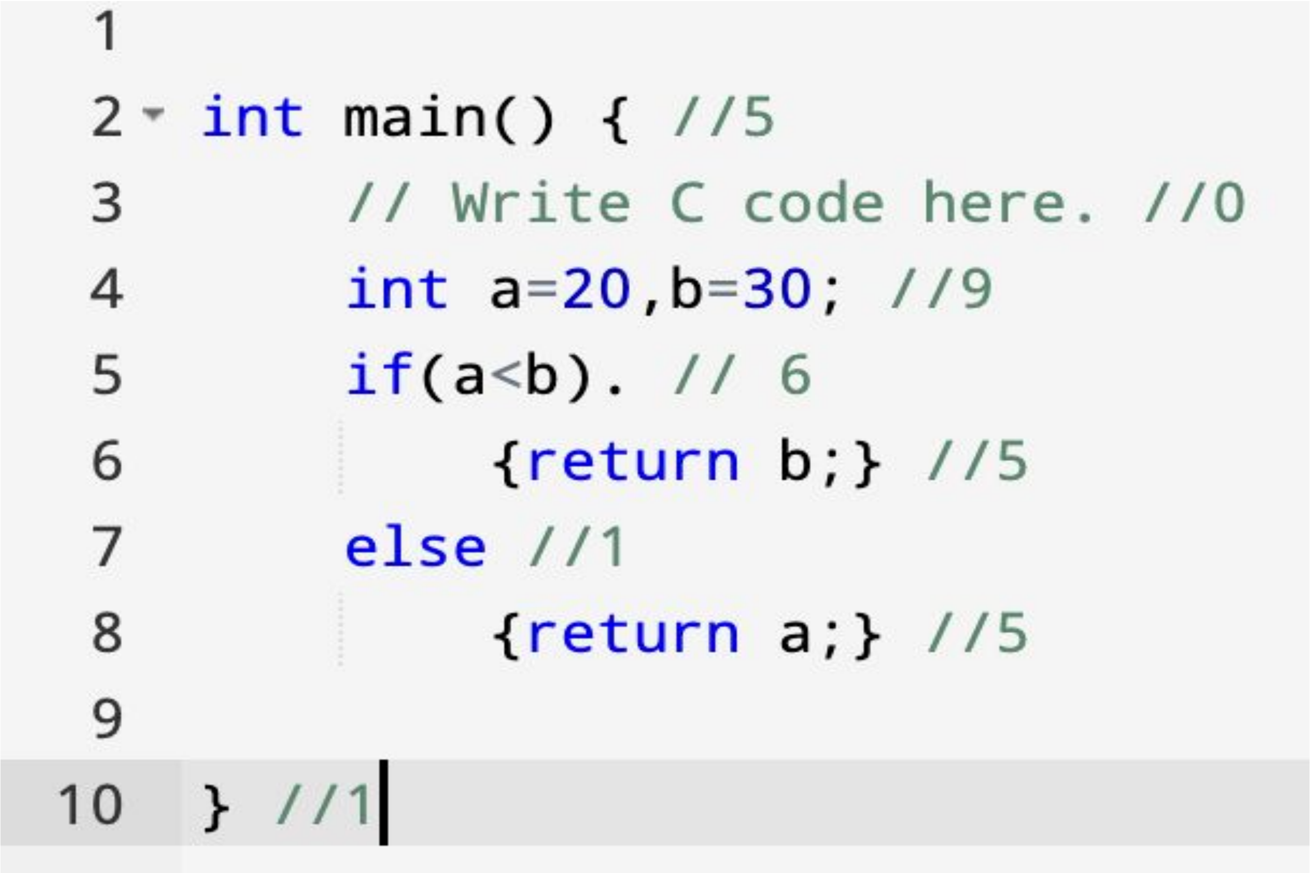
Lexical Analyzer



Lexical Analyzer

- Lexical analyzer uses DFA to do tokenization.
- While doing tokenization lexical analyzer always gives importance to longest matching.

Counting no of tokens (Ex: 1)



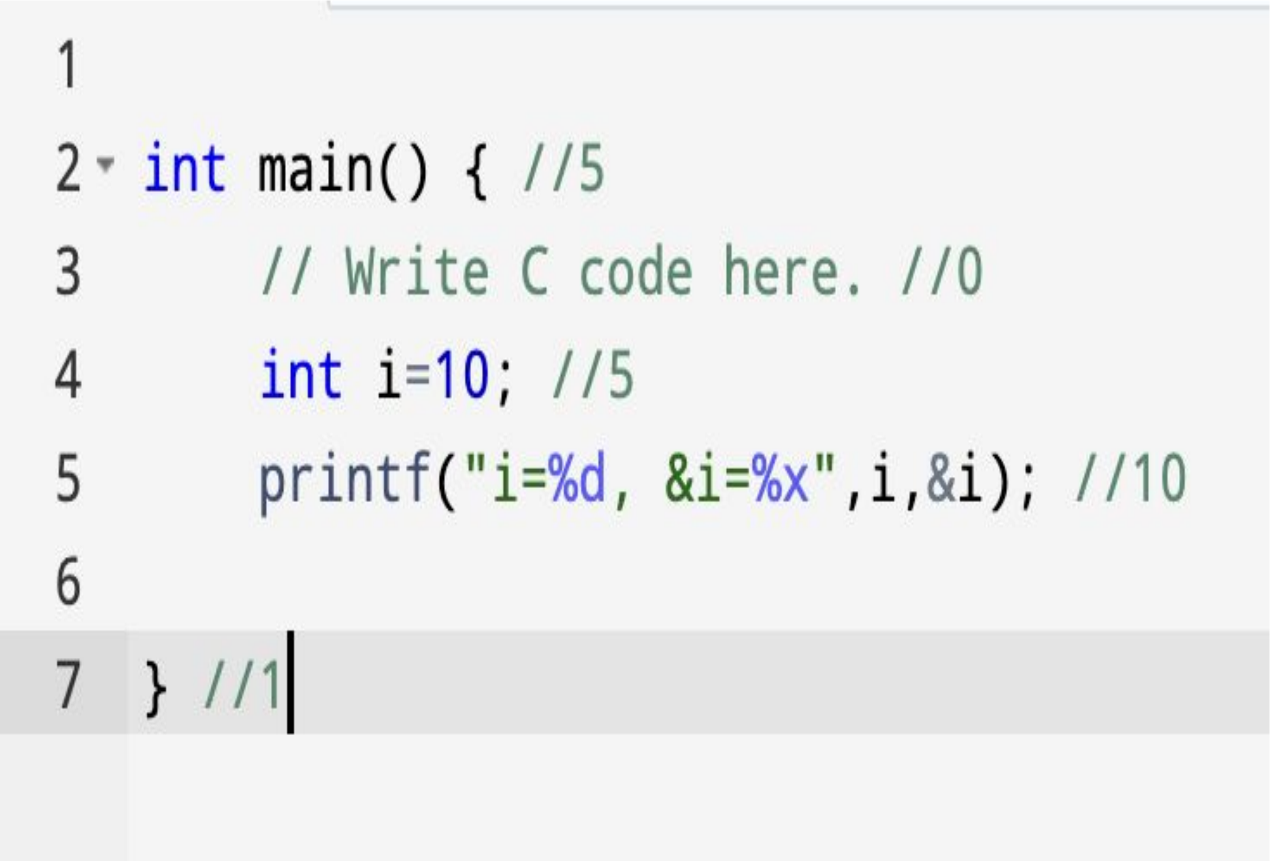
```

int main() { //5
    // Write C code here. //0
    int a=20,b=30; //9
    if(a<b). // 6
        {return b;} //5
    else //1
        {return a;} //5

} //1
```

Total tokens = 32

Counting no of tokens (Ex: 2)



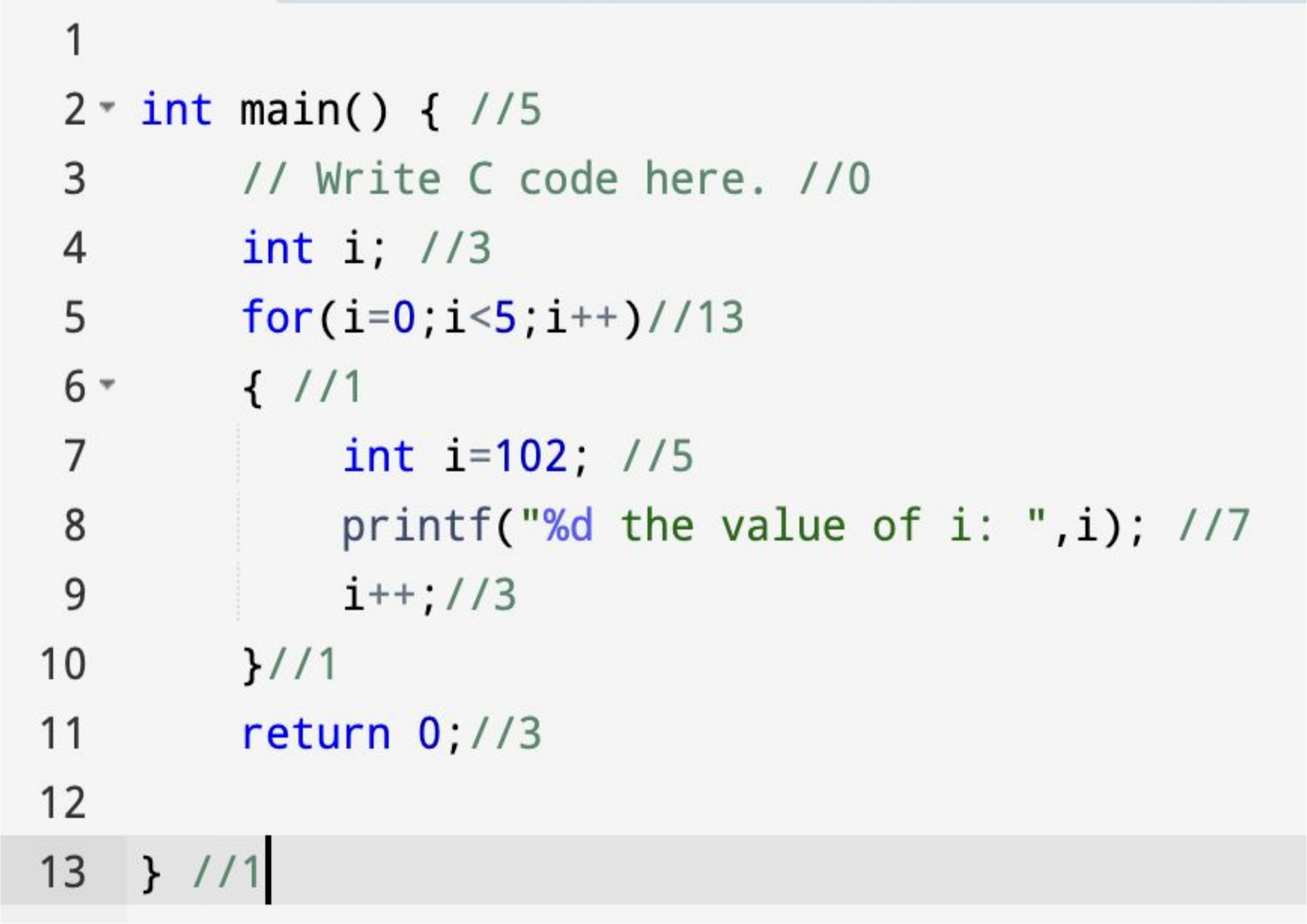
```

int main() { //5
    // Write C code here. //0
    int i=10; //5
    printf("i=%d, &i=%x",i,&i); //10

} //1
```

Total tokens = 21

Counting no of tokens (Ex: 3)



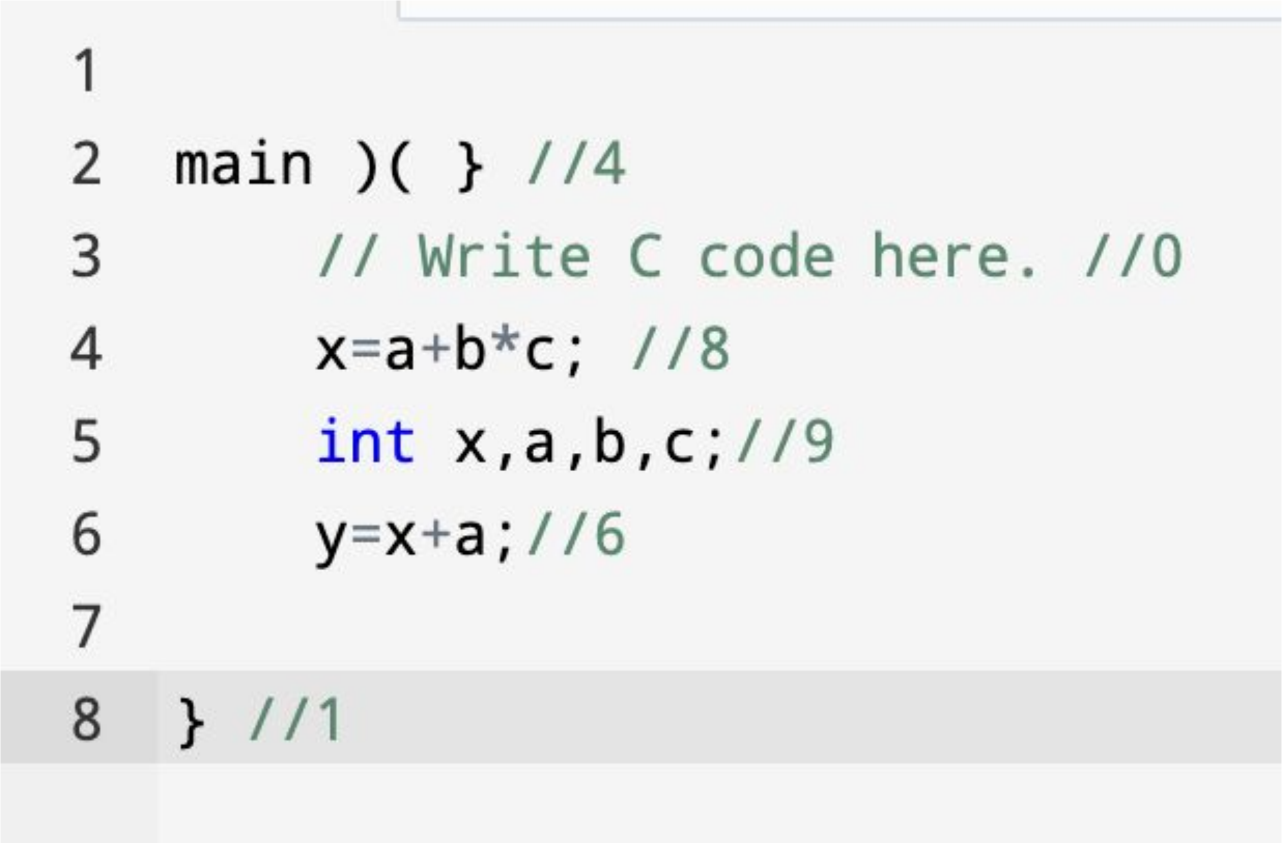
```

int main() { //5
    // Write C code here. //0
    int i; //3
    for(i=0;i<5;i++)//13
    { //1
        int i=102; //5
        printf("%d the value of i: ",i); //7
        i++;//3
    }//1
    return 0;//3

} //1
```

Total tokens = 42

Counting no of tokens (Ex: 4)

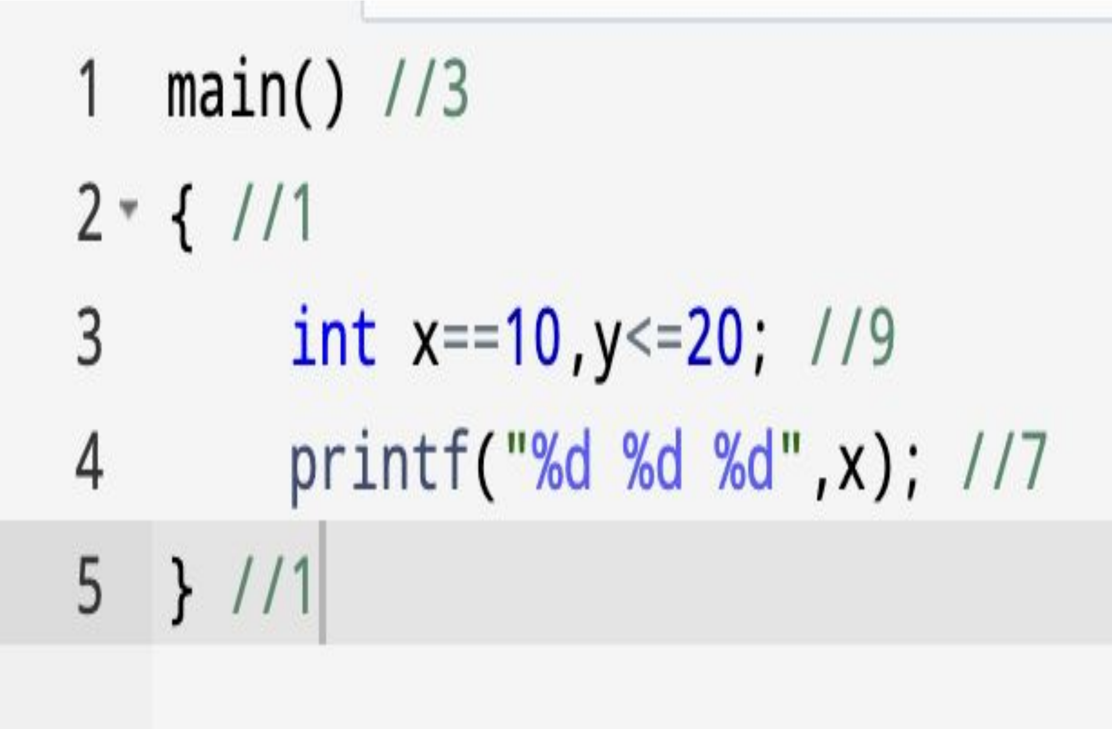


```
main )( } //4
    // Write C code here. //0
    x=a+b*c; //8
    int x,a,b,c;//9
    y=x+a;//6

} //1
```

Total tokens = 28

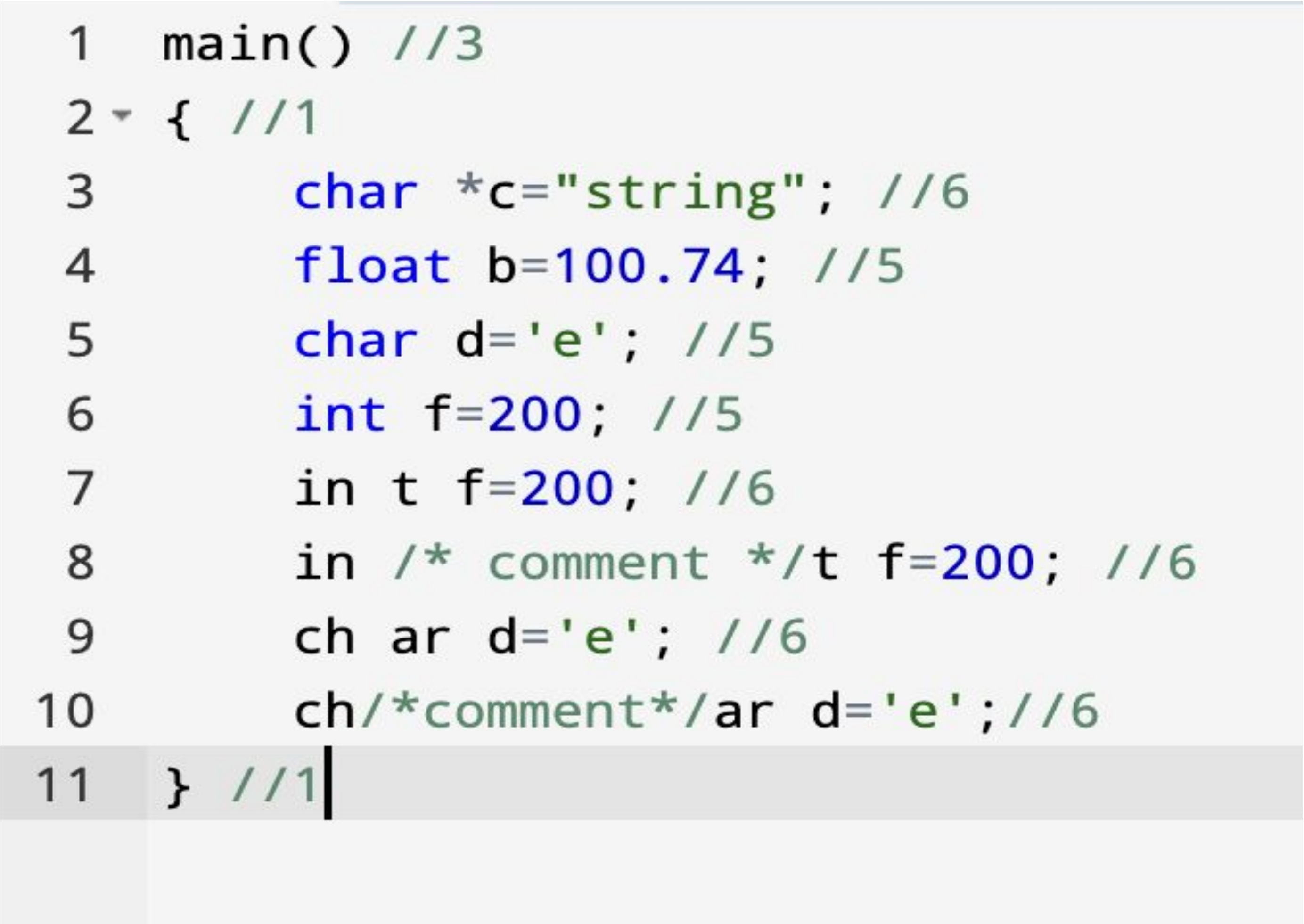
Counting no of tokens (Ex: 5)



```
main() //3
{ //1
    int x==10,y<=20; //9
    printf("%d %d %d",x); //7
} //1
```

Total tokens = 21

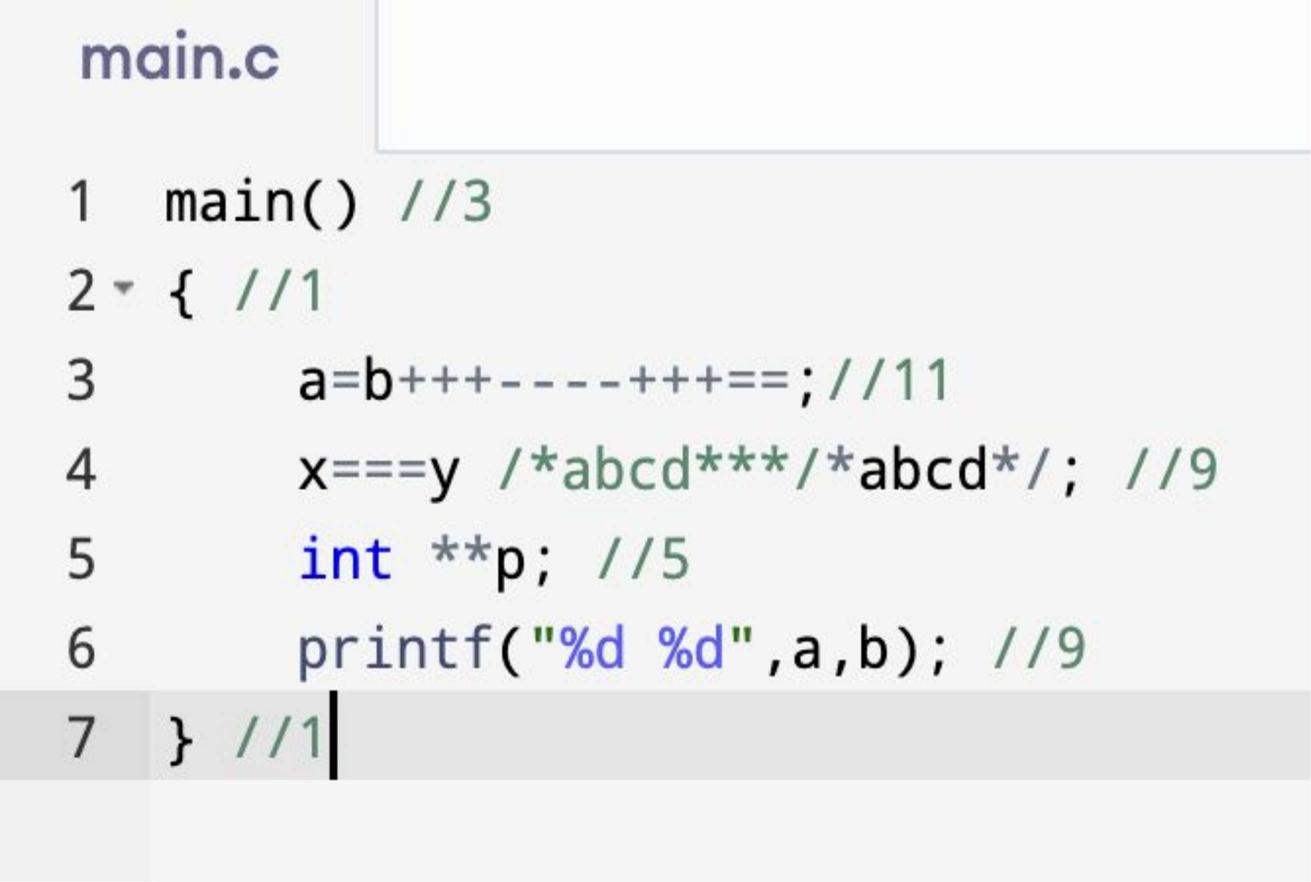
Counting no of tokens (Ex: 6)



```
main() //3
{ //1
    char *c="string"; //6
    float b=100.74; //5
    char d='e'; //5
    int f=200; //5
    in t f=200; //6
    in /* comment */t f=200; //6
    ch ar d='e'; //6
    ch/*comment*/ar d='e';//6
} //1
```

Total tokens = 50

Counting no of tokens (Ex: 7)



```c
main() //3
{ //1
    a=b+++----+++==;//11
    x===y /*abcd***/*abcd*/; //9
    int **p; //5
    printf("%d %d",a,b); //9
} //1
```

Total tokens = 39

Errors Detected in Lexical analysis

- Numeric literals that are too long. (int a=1234567891011121314151617181920)
- Long identifiers
- Ill-formed numeric literals. (int a= $123)
- Input characters that are not in the source language.

Error Recovery

- **Delete:** Unknown characters are deleted. Also known as panic mode recovery. Example:
  - ‘Charr’ corrected as ‘char’ deleting r.
- **Insert:** An extraa or missing charactered is inserted to form a meaningful token.Example:
  - ‘cha’ corrected as ‘char’ inserting ‘r’
- **Transpose:** Based on certain rules we can transpose two characters. Example:
  - ‘Whiel’ can be corrected to ‘while’

Error Recovery

- **Replace**: Based on replacing one character by another. Example:
  - ‘chrr’ can be corrected as ‘char’ by replacing ‘r’ with ‘a’.