Token, white space, Comments, variable, Parser

Subject: C language Date: 21 / 12 / 2025

1. What is Tokens ?

⇒ A token is the smallest indivisual unit of a program that the compiler recognizes while Converting source code into machine code.

- The text of C/C++ program Consists of tokens and white space.

2. What is Parser ?

⇒ Parser is a program that analyzes the sequence of tokens to check whether they follow the grammatical rules of the language.

OR

Parser checks the syntax of a program using tokens generated by the lexical analyzer.

3. What is Lexical Analyzer ?

⇒ A lexical analyzer is the first phase of a Compiler that reads the source code character by character and Converts it into tokens.

Subject: ________ Date: __/__/____

4. Type of Tokens ?

⇒ 1.) keyword
- Reserved words with predefined meaning.
- Cannot be used as identifiers.

Example :— int, float, if, else, while, for, return.

2.) Identifiers
- Names given to variables, functions, array etc.
- Defined by the programmer.

Rules :
- Must start with a letter or underscore (_).
- Cannot be a keyword

Example :— Sum, total_marks, _Count

3.) Constants
- Fixed values that do not change.

Two types of Constants :—

1) Primary Constants

i.) Integer Constants.

Example :— 10, -5, 100.

ii.) Floating (Real) Constants.

Example :— 3.14, 2.5

Subject: ________ Date: __/__/__

iii.) Character Constants.

Example :— 'A', '9'

NOTE :— Single digit ya single character ko jab hum single quote me likhte hai, tab use hum character Constant kahte hain.

2.) Secondary Constants

i.) Array Constants

Example :— {1, 2, 3, 4}

ii) String Constants

Example :— "Hello"

iii) Pointer Constants

Example :— Address values like &a

iv) Structure Constants

v) Union Constants

vi) Enumerator Constants

4.) Operators

- Symbols used to perform operations.

Example :— +, -, *, /, %, ==, >=, <=, >, <, &&, ||. etc

5.) Punctuators
- Symbols that separate or structure the program.

Example :- ; , ( ) { } [ ]

6.) String Literals
- Sequence of characters inside double quotes.

Example :- " Computer Science"

5. What is white Space ?

⇒ Whitespace refers to blank characters that are used to separate tokens and improve readability of the program.

Example :- Tabs (\t)
New lines (\n)
Blanks (' ')
Form Feed (\f)

6. Is whitespace Compulsory for separating tokens in C ?

⇒ Whitespace is not always ~~nesso~~ necessary for token separation. The Compiler Can Separte tokens using operators and punctuators, but whitespace is required where tokens may Combine and change meaning.

Subject: ______ Date: __/__/__

7. ## —: Comments :—

- Comments are text ignored by Compiler.
- Comments are useful for documentation of your Code and useful for programmers.

There are two types of Comments :—

i) Single line Comment

e.g; // Hello world

ii) Multiple line Comment

```
e.g;   /*
            Hello world
            Hello world 2
       */
```

8. What is variable's ?

→ Variable is a name of memory location where we store program's data during execution of program.

- Variable name is any Combination of alphabets (a to z or A to Z), digits (0 to 9) and underscore(_).

- No other symbol is allowed.

- valid variable name Cannot start from a digit.

- Variable name Cannot be a Keyword.

9. ## = : Keywords in C : —

auto	Const	double	float	int	Short
break	Continue	else	for	long	Signed
Case	default	enum	goto	register	sizeof
char	do	extern	if	return	Static
struct	Switch	typedef	union	unsigned	void
volatile	while				

C89 Keywords (32)

_Bool
_Complex
_Imaginary
inline
restrict

C99 Keywords (5)

_Alignas
_Alignof
_Atomic
_Generic
_Noreturn
_Static_assert
_Thread_local

C11 Keywords(7)

10. Why we need Data Classification in C.

⇒ Data classification means grouping data into different data types such as int, float, char, double, etc.

1. Efficient Memory Utilization.

⇒ Different types of data require different amounts of memory. By classifying data, C allocates only the required memory.

Subject: Date: / /

Example :-
- int → 4 bytes
- char → 1 byte
- double → 8 bytes

This avoids wasting memory.

2. Correct Interpretation of Data.

⇒ The same binary can represent different values depending on its type.
Data classification tells the Compiler how to interpret the data.

Example :-
- 65 as int → number 65
- 65 as char → character 'A'.

3. Performing Valid Operations.

⇒ Certain operations are valid only for specific data types

Example :-
- Arithmetic Operations → int, float
- Character Operations → char
- Logical Operations → integer types

Data classification prevents invalid operations.

4. Improves Program Efficiency and Speed:

⇒ knowing the data type allows the Compiler to generate optimized machine Code, making programs faster and more efficient.

5. Error Detection and Debugging.

⇒ If data is not properly classified, the Compiler Can detect type errors during Compilation.

Example :-

```
    int x;
    x = 3.5; // warning or data loss
```

This helps in finding bugs early.

6. Better Readability and Maintainability.

⇒ Using proper data types makes programs easy to understand, modify, and maintains.

Example :-

```
 int age;
 float salary;
 char grade;
```

The purpose of each variable becomes clear.

Subject: ______ Date: / /

7. Supports Complex Data Handling.

⇒ Data classification enables advanced data handling using:

- Arrays
- Structures
- Unions
- Pointers

These are essential for large and Complex programs.

II. Factors Responsible for Data Classification.

1. Nature of Data

⇒ The type of value to be stored decides the data classification.

Example :-

- Whole numbers ⟶ int
- Decimal numbers ⟶ float, double
- Single characters ⟶ char

2. Memory Requirement

⇒ Different data types require different amounts of memory space. Data is classified to allocate minimum and appropriate memory.

Subject: ______ Date: __/__/____

Example :—
- char → 1 byte
- int → 4 bytes

3. Range of Values

⇒ Each data type has a fixed range of value it can store.
Data classification ensures the data fits within the required range.

Example :—
- int → -32,768 to 32,767 (approx, depends on system)
- float → very large range.

4. Precision Required

⇒ Some data needs exact values, while others need fractional precision.

Example :—
- Age → int
- Salary → float
- Scientific Calculations → double

5. Type of operations to be Performed.

⇒ The operations to be applied to data influence classification.

Subject: Date: / /

Example :-

- Arithmetic Calculation's ⟶ numeric types
- Character Comparison ⟶ char
- Logical decisions ⟶ integer types

6. Speed and Efficiency

⇒ Smaller data types are faster to process and use less memory. Data classification helps improve execution speed.

7. Program Readability and Maintenance.

⇒ Using appropriate data types makes programs clear, structured, and easy to maintain.

12. What is Data Types?

⇒ Data types specify the type of data, memory size, and operations that Can be performed on a variable.

13. Types of Data Type ?

1. Basic (Primitive) Data Types.

⇒ used to store simple values.

- int — stores integers

- float — stores decimal numbers.
- double — stores large decimal numbers with high precision.
- char — stores a single character.

2. Derived Data Types

⇒ Derived from basic data types.
- Array — Collection of similar data types.
- Pointer — stores address of a variable.
- Structure — Collection of different data types.
- Union — shares memory among different data types.

3. Enumeration (enum)

⇒ Used to define a set of named integer Constants.

4. Void Data Type.
- void — represents no value or no data type.

14. What is variable Declaration.

⇒ Variable decleration is the process of declaring a variable with a data type and name so that the Compiler knows what type of data it will store and how much memory to allocate.

Example :- data_type variable_name;

NOTE:- 1 bytes = 8 bits.

Subject: ________ Date: __/__/__

```
int age;
float salary;
char grade;
```

15. —— : float vs double :——

i) float 4 bytes leta hai and double 8 bytes leta hai memory.

ii) float less accurate value store karta hai & double more accurate value store karta hai.