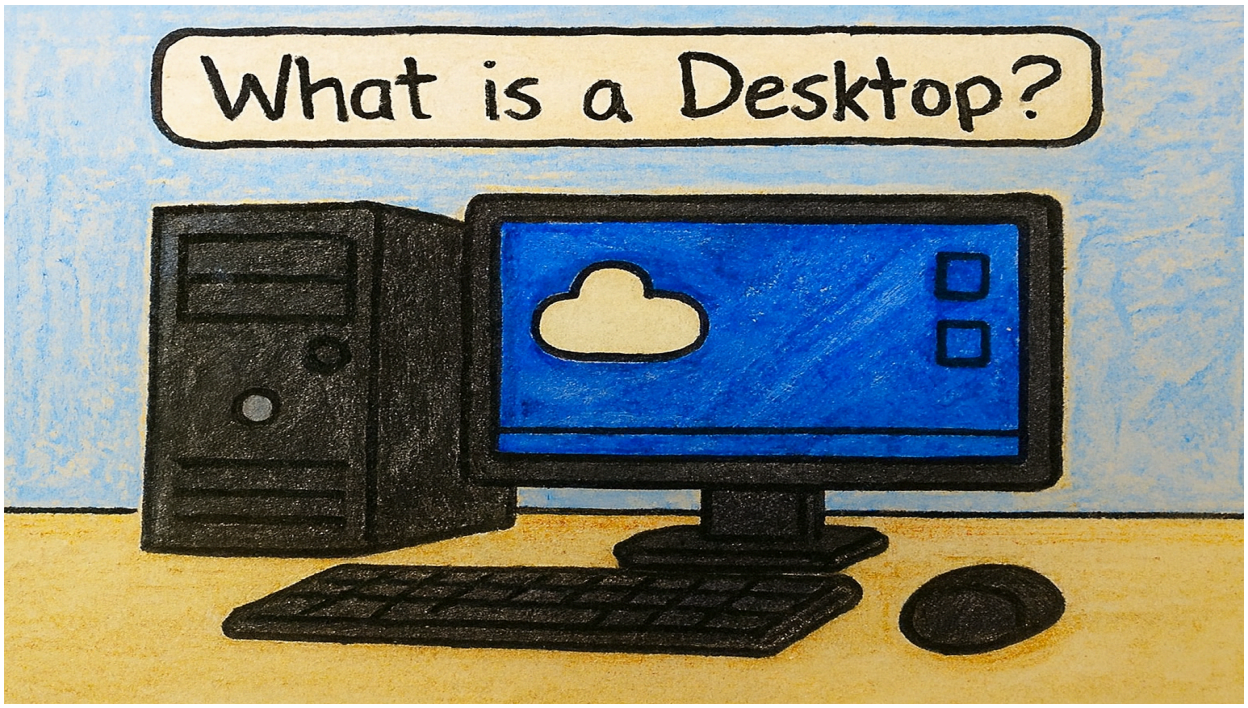

Class: 12th

Subject: Computer

Chapter 9: ELEMENTS OF C

📍 Important MCQs:

1. A computer can perform tasks only when it is given:

(a) Electricity

(b) Hardware

(c) Proper instructions (programs) ✅

(d) Memory

2. C language is used to:

(a) Write documents

(b) Solve multidimensional problems ✅

(c) Design graphics

(d) Store data only

3. The basic elements of C language are called:

(a) Statements

(b) Functions

(c) Building blocks of a C program ✅

(d) Operators

4. An identifier is:

(a) A keyword

(b) A constant

(c) A name used to represent program elements ✅

(d) A data type

5. Which of the following can be an identifier in C?

(a) Variable

(b) Function

(c) Label

(d) All of these ✅

6. Only the first ___ characters of an identifier are significant in C.

(a) 20

(b) 25

(c) 31 ✅

(d) 40

7. The two types of identifiers in C are:

(a) Local and global

(b) Integer and float

(c) Standard and user-defined ✅

(d) Keywords and constants

8. Which of the following is a standard identifier?

(a) total

(b) value

(c) printf ✅

(d) result

9. Redefining a standard identifier is:

(a) Recommended

(b) Necessary

(c) Not recommended ✅

(d) Compulsory

10. C is a case-sensitive language. It means:

(a) Only uppercase letters are allowed

(b) Only lowercase letters are allowed

(c) Uppercase and lowercase are treated differently ✅

(d) Case does not matter

11. Which pair of identifiers is different in C?

(a) sum and sum

(b) count and count

(c) Area and area ✅

(d) total and total

12. Keywords in C are also called:

(a) Identifiers

(b) Variables

(c) Reserved words ✅

(d) Constants

13. How many keywords are there in ANSI C?

(a) 30

(b) 31

(c) 32 ✅

(d) 33

14. Keywords are always written in:

(a) Uppercase

(b) Lowercase ✅

(c) Capital letters

(d) Mixed case

15. Keywords in C:

(a) Can be redefined

(b) Can be used as variables

(c) Cannot be used for any other purpose ✅

(d) Are optional

16. A variable is:

(a) A fixed value

(b) A named memory location ✅

(c) A keyword

(d) An operator

17. Variables store data temporarily in:

(a) ROM

(b) Hard disk

(c) RAM ✅

(d) Cache

18. C is a strongly typed language because:

(a) It has many keywords

(b) Variables must be declared before use ✅

(c) It is case-sensitive

(d) It uses symbols

19. Assigning a value to a variable at the time of declaration is called:

(a) Declaration

(b) Definition

(c) Initialization ✅

(d) Compilation

20. Which of the following is a valid variable name?

(a) 9marks

(b) #total

(c) interest_rate ✅

(d) problem solving

21. A constant in C is a quantity whose value:

(a) Changes during execution

(b) Can be modified by user

(c) Cannot be changed during program execution ✅

(d) Is stored temporarily

22. Which directive is used to define a constant macro in C?

(a) #include

(b) #define ✅

(c) #constant

(d) #macro

23. The statement #define PI 3.142857 defines PI as:

(a) A variable

(b) A function

(c) A constant macro ✅

(d) A keyword

24. How many types of constants are there in C?

(a) One

(b) Two ✅

(c) Three

(d) Four

25. Which of the following is NOT a type of constant in C?

(a) Numeric constant

(b) Character constant

(c) String constant

(d) Logical constant ✅

26. Integer constants are those numbers which:

(a) Have fractional parts

(b) Are measured

(c) Do not have fractional parts ✅

(d) Are written in scientific notation

27. Which of the following is a floating-point constant?

(a) 45

(b) -12

(c) 5.0 ✅

(d) +78

28. A character constant must be enclosed within:

(a) Double quotation marks

(b) Brackets

(c) Apostrophes (single quotes) ✅

(d) Curly braces

29. Which of the following is a valid character constant?

(a) "A"

(b) 'AB'

(c) '5' ✅

(d) 5

30. The maximum length of a character constant in C is:

(a) 2 characters

(b) 4 characters

(c) 1 character ✅

(d) Unlimited

31. A data type in C defines:

(a) Program structure

(b) Memory address

(c) A set of values and operations on those values ✅

(d) Program output

32. Which of the following is a standard data type in C?

(a) number

(b) string

(c) int ✅

(d) logical

33. Which data type is used to store whole numbers without fractional parts?

(a) float

(b) double

(c) int ✅

(d) char

34. How many bytes of memory are allocated to an int variable in C?

(a) 1 byte

(b) 2 bytes ✅

(c) 4 bytes

(d) 8 bytes

35. The range of values for a signed int data type is:

(a) 0 to 65535

(b) −128 to 127

(c) −32768 to 32767 ✅

(d) −2147483648 to 2147483647

36. Operators in C are symbols which are used to:

(a) Store data

(b) Define variables

(c) Perform operations on data ✅

(d) Control program flow

37. Which of the following is an arithmetic operator in C?

(a) &&

(b) ==

(c) % ✅

(d) ||

38. The modulus operator (%) is used to:

(a) Find quotient

(b) Find remainder of an integral division ✅

(c) Multiply two numbers

(d) Compare two values

39. If a = 8 and b = 3, then the value of a % b will be:

(a) 1

(b) 2 ✅

(c) 3

(d) 5

40. Relational operators in C always evaluate to:

(a) Integer values only

(b) Characters

(c) True or False (non-zero or zero) ✅

(d) Floating point values

41. Which of the following is a relational operator?

(a) +

(b) &&

(c) >= ✅

(d) =

42. Logical AND (&&) operator evaluates to true when:

(a) Any one condition is true

(b) Both conditions are true ✅

(c) Both conditions are false

(d) Only first condition is true

43. Logical OR (||) operator evaluates to false when:

(a) Both conditions are true

(b) Any one condition is true

(c) Both conditions are false ✅

(d) First condition is false

44. The logical NOT (!) operator:

(a) Combines two conditions

(b) Reverses the result of a condition ✅

(c) Compares two values

(d) Assigns a value

45. The assignment operator in C is represented by the symbol:

(a) ==

(b) :=

(c) = ✅

(d) =>

46. An expression in C is:

(a) A single variable

(b) A constant only

(c) A combination of operators and operands ✅

(d) A keyword

47. Which of the following is an arithmetic expression?

(a) a > b

(b) a && b

(c) a + b ✅

(d) a = b

48. If both operands of an expression are of type int, the result will be of type:

(a) float

(b) double

(c) char

(d) int ✅

49. What will be the result of the expression 7 / 2 in C?

(a) 3.5

(b) 4

(c) 3 ✅

(d) 2

50. Which symbol is used to start a single-line comment in C?

(a) /*

(b) */

(c) // ✅

(d) ##

📍 Important Short Questions:

1. What is a computer program?

Answer:

👉 A computer program is a set of instructions written in a programming language that tells the computer how to perform specific tasks.

Example: A program to calculate the area of a rectangle.

2. What is an identifier in C?

Answer:

👉 An identifier is a name used to represent variables, constants, functions, types, or labels in a C program.

Example: length, totalMarks, count.

3. What are standard identifiers?

Answer:

👉 Standard identifiers are predefined names in C with special meanings, usually used in the standard library.

Example: printf, scanf.

4. What are user-defined identifiers?

Answer:

👉 User-defined identifiers are names created by the programmer to access memory locations or store data in a program.

Example: score, height, totalStudents.

5. Is C case-sensitive? Explain.

Answer:

👉 **Yes**, C is case-sensitive, meaning uppercase and lowercase letters are treated as different characters.

Example: Length and length are considered two different variables.

6. What is a keyword in C?

Answer:

👉 A keyword is a reserved word in C with a predefined meaning that cannot be used as a variable name.

Example: int, float, if, while.

7. What is a variable?

Answer:

👉 A variable is a named memory location used to store data that can change during program execution.

Example: int age = 20; (here age is a variable storing 20).

8. How do you declare a variable in C?

Answer:

👉 A variable is declared by specifying its data type followed by its name.

Example: int count; or double length;.

9. What is variable initialization?

Answer:

👉 Initialization is the process of assigning a value to a variable at the time of declaration.

Example: int count = 100; or float weight = 75.8;.

10. List some rules for naming variables in C.

Answer:

👉 **Rules for naming variables:**

- Can contain letters, digits, and underscores _.

- First character must be a letter (underscore allowed but not recommended).
- Cannot use keywords as variable names.
- Cannot contain spaces.
- Case-sensitive.
- **Example:** totalMarks, interestRate, score1.

11. What is a constant in C?

Answer:

👉 A constant is a quantity whose value cannot be changed during program execution.

Example: #define PI 3.142857

12. What are numeric constants?

Answer:

👉 Numeric constants are numbers whose values do not change. They can be integers or floating-point numbers.

Example: 56, -678, 4.786, 0.45

13. What is a character constant?

Answer:

👉 A character constant is a single alphabet, digit, or symbol enclosed in apostrophes. Its length is 1 character.

Example: 'A', '5', '='

14. What is a data type in C?

Answer:

👉 A data type defines a set of values and the operations that can be performed on them. It determines what type of data a variable can store.

Example: int, float, char

15. Name the standard integer data types in C.

Answer:

👉 The standard integer data types are: int, short int (short), long int (long).

16. What is the difference between signed and unsigned integers?

Answer:

👉 Signed integers can store both positive and negative numbers, whereas unsigned integers can store only positive numbers (including zero).

Example: signed int a = -10; unsigned int b = 10;

17. What are floating-point data types in C?

Answer:

👉 Floating-point data types store numbers with fractional parts. They include: float, double, long double.

Example: float weight = 75.8; double price = 245.67;

18. Explain the representation of floating-point numbers in C.

Answer:

👉 Floating-point numbers are stored in mantissa (value) and exponent (power of 10) form. Scientific notation is used: 2.45634e5 represents 245634.

19. What is a char data type in C?

Answer:

👉 char is used to store a single character, occupying 1 byte of memory. Characters are represented using ASCII codes.

Example: 'a', '5', '#'

20. What is arithmetic overflow and underflow in C?

Answer:

👉 **Arithmetic overflow** occurs when a number is too large to be represented accurately.

👉 **Arithmetic underflow** occurs when a number is too small to be represented accurately and may be stored as zero.

Example: Adding 1970.0 + 0.0000001243 may result in 1970.0 due to cancellation error.

21. What is an operator in C?

Answer:

👉 An operator is a symbol used to perform operations on data, such as arithmetic, relational, or logical operations.

Example: +, -, *, /, %

22. What are arithmetic operators in C?

Answer:

👉 Arithmetic operators are used to perform arithmetic calculations like addition, subtraction, multiplication, division, and modulus.

Example: +, -, *, /, %

23. What is the modulus operator (%) in C?

Answer:

👉 The modulus operator returns the remainder of integer division.

Example: 8 % 3 = 2

24. What are relational operators?

Answer:

👉 Relational operators compare two values and return true (1) or false (0).

Example: >, <, >=, <=, ==, !=

25. What are logical operators?

Answer:

👉 Logical operators combine relational expressions to form compound conditions. They include:

- && (AND)
- || (OR)
- ! (NOT)

Example: salary < 1000 && status == 'M'

26. What is an assignment operator?

Answer:

👉 The assignment operator = is used to assign a value or expression result to a variable.

Example: Area = height * width;

27. Explain the increment operator in C.

Answer:

👉 The increment operator ++ increases a variable's value by 1. It can be prefix (++i) or postfix (i++).

Example:

j = 10; i = ++j; ⟶ i = 11, j = 11

j = 10; i = j++; ⟶ i = 10, j = 11

28. Explain the decrement operator in C.

Answer:

👉 The decrement operator -- decreases a variable's value by 1. It can be prefix (--i) or postfix (i--).

Example:

j = 10; i = --j; ⟶ i = 9, j = 9

j = 10; i = j--; ⟶ i = 10, j = 9

29. What are compound assignment operators?

Answer:

👉 Compound assignment operators combine arithmetic operations with assignment. Examples: +=, -=, *=, /=.

Example:

- j = 10; j += 5; ⟶ j = 15
- j = 10; j /= 5; ⟶ j = 2

30. What is operator precedence in C?

Answer:

👉 Operator precedence determines the order in which operators are evaluated in an expression. Higher precedence operators are evaluated first.

Example: In a + b * c, multiplication (*) is performed before addition (+).

31. What is an expression in C?

Answer:

👉 An expression is a combination of operators and operands. Operands may be constants or variables.

Example: a + b, 7 + m

32. What is an arithmetic expression?

Answer:

👉 An arithmetic expression is an expression in which only arithmetic operators are applied to operands.

Example: x * y + z / 2

33. How is the data type of an expression determined?

Answer:

👉 The data type of an expression depends on the data types of its operands.

- If both operands are int, the result is int.
- If operands are of mixed types (e.g., int and double), the result is double.

34. How does C handle division of integers?

Answer:

👉 When both dividend and divisor are integers, C performs integer division and truncates the fractional part.

Example:

7 / 2 = 3 (fractional part discarded)

To get a precise result, at least one operand must be a floating-point number:

7.0 / 2 = 3.5

35. What are comments in C and their types?

Answer:

👉 Comments are notes added to a program to improve readability and help in debugging. They are ignored by the compiler.

Types:

Single-line comment: starts with //

Example: // This program calculates factorial

Multi-line comment: starts with /* and ends with */

Example:

/* This program calculates

the sum of two numbers */

📍 Exercise 9c

1. Fill in the blanks:

(i) The first character of a variable name must be a _____.

Answer: Letter

Explain: In C, a variable name must begin with a letter to be valid. Starting with a number or special character is not allowed.

Example: age, height

(ii) _____operates on two operands.

Answer: Binary operator

Explain: A binary operator is an operator that works on two operands, e.g., +, -, *, /.

Example: a + b, x * y

(iii) Named memory cells which are used to store program's input and output are called_____.

Answer: Variables

Explain: Variables are memory locations that store data temporarily during program execution.

Example: int count;, float price;

(iv) The maximum length of a character constant is _____ character.

Answer: 1

Explain: A character constant can store only a single character enclosed in single quotes.

Example: 'A', '9'

(v) The value of a variable of type int ranges from _____ to_____.

Answer: -32768 to 32767

Explain: int type variables in C can store signed integers in this range (16-bit memory).

Example: int a = 100;

(vi) The value of an unsigned int variable ranges from 0 to _____.

Answer: 65535

Explain: Unsigned int stores only positive integers and zero in 16-bit memory.

Example: unsigned int u = 5000;

(vii) The value of a float variable ranges from _____ to _____.

Answer: 3.4E-38 to 3.4E+38

Explain: Float variables store decimal numbers in scientific notation format with limited precision.

Example: float f = 12.345;

(viii) The symbol for logical OR operator is _____.

Answer: ||

Explain: The logical OR operator returns true if any of the conditions is true.

Example: if(a>5 || b<10)

(ix) _____ are used to increase the readability of the program.

Answer: Comments

Explain: Comments are notes inserted in the code to help programmers understand it better.

Example: // This is a single line comment

(x) Multi-line comments start with _____.

Answer: /*

Explain: Multi-line comments start with /* and end with */. They can span multiple lines.

Example:

/* This program calculates

the sum of two numbers */

2. Choose the correct option:

(i) Variables are created in:

(a) RAM ✅

(b) ROM

(c) Hard Disk

(d) Cache

(ii) Which of the following is a valid character constant?

(a) a

(b) "b"

(c) '6' ✅

(d) =

(iii) Which of the following data type offers the highest precision?

(a) float

(b) long int

(c) long double ✅

(d) unsigned long int

(iv) When the result of the computation of two very small numbers is too small to be represented, this phenomenon is called:

(a) Arithmetic overflows

(b) Arithmetic underflow ✅

(c) Truncation

(d) Round off

(v) The symbol '=', represents:

(a) Comparison operator

(b) Assignment operator ✅

(c) Equal-to operator

(d) None of these

(vi) Which of the following operators has lowest precedence?

(a) !

(b) +

(c) = ✅

(d) ==

(vii) Relational operators are used to:

(a) Establish a relationship among variables

(b) Compare two values ✅

(c) Construct compound condition

(d) Perform arithmetic operations

(viii) C is a strongly typed language, this means that:

(a) Every program must be compiled before execution

(b) Every variable must be declared before it is being used ✅

(c) The variable declaration also defines the variable

(d) Sufficient data types are available to manipulate each type of data

(ix) The logical not operator, denoted by !, is a:

(a) Ternary operator

(b) Unary operator ✅

(c) Binary operator

(d) Bitwise operator

(x) a += b is equivalent to:

(a) b += a

(b) a =+ b

(c) a = a + b ✅

(d) b = b + a

3. Write T for true and F for false statement.

(i) printf and scanf are standard identifiers.

Answer: T ✅

(ii) In C language, all variables must be declared before being used.

Answer: T ✅

(iii) Standard data types are not predefined in C language.

Answer: F ❌ (They are predefined)

(iv) The double data type required 4 bytes in memory.

Answer: F ❌ (It usually requires 8 bytes)

(v) In Scientific notation, the exponent represents the value of the number and mantissa represents the power to which it is raised.

Answer: F ❌ (Mantissa is the value, exponent is the power)

(vi) The symbol for modulus operator is %.

Answer: T ✅

(vii) The symbol = is used to compare two values.

Answer: F ❌ (= is assignment operator, == is for comparison)

(viii) Operator precedence determines the order of evaluation of operators in an expression.

Answer: T ✅

(ix) For many compilers a C variable name can be up to 31 characters.

Answer: T ✅

(x) C program can only use lowercase letters in variable names.

Answer: F ❌ (Both uppercase and lowercase letters can be used)

🌟 Q.4: What is an identifier? Discuss the two types of identifiers in C.

❖ **Answer:**

An identifier is a name used in a C program to represent variables, constants, functions, types, or labels. Identifiers act as labels that allow the programmer to access and manipulate data stored in memory. They help the compiler and the programmer to identify and work with different memory locations.

C identifiers must follow certain rules:

- They can contain letters, digits, and the underscore (_) character.
- They must begin with a letter (or underscore, though not recommended).
- C is case sensitive, which means count and COUNT are treated as two different identifiers.

- Keywords cannot be used as identifiers.

Types of Identifiers in C:

1. Standard Identifiers:

- These are predefined in C and have special meanings.
- Examples include standard library functions like printf() and scanf().
- They can be redefined by the programmer, but it is not recommended because it can lead to unexpected results.

2. User-defined Identifiers:

- These are defined by the programmer to name variables, functions, constants, or labels.
- They help the programmer access specific memory locations to store or retrieve data.

Examples: length, totalMarks, SquareArea.

✦ Summary:

Identifiers are names that give meaning to memory locations in a program. C provides standard identifiers (predefined) and user-defined identifiers (defined by the programmer). Proper

use of identifiers improves program readability and helps in organizing data efficiently.

🌟 Q.5: What is a variable? Discuss the difference between declaring and defining a variable.

❖ **Answer:**

A variable is a named memory location in a computer's RAM that is used to store data temporarily during the execution of a program. The value of a variable can change during program execution, which is why it is called a "variable."

For example:

C

```
int age;

age = 20;
```

Here, age is a variable of type int and it stores the value 20.

Declaring vs Defining a Variable:

1. Declaring a Variable:

- Variable declaration tells the compiler the name and type of a variable that will be used in the program.

- It does not necessarily allocate memory.
- **Syntax:**

C

```
int marks;  // declaration of variable 'marks' of type int
```

- **Purpose:** The compiler knows that marks is an integer, but memory may or may not be allocated yet depending on the system.

2. Defining a Variable:

- Variable definition tells the compiler to allocate memory for that variable.
- Often in C, declaration also acts as definition.
- **Syntax:**

C

```
int marks;  // declaration + definition (memory allocated)
```

- If we assign a value during definition, it is also called initialization:

C

```
int marks = 85;  // declaration + definition + initialization
```

✦ **Summary:**

- A variable is a memory location to store data that can change.
- Declaration tells the compiler the variable's name and type.
- Definition allocates memory for the variable.
- Initialization assigns a starting value to the variable.

🌟 **Q.6: Write down rules for naming variables in C.**

❖ **Answer:**

In C programming, a variable is a name given to a memory location used to store data. But we cannot give any random name; the name of a variable must follow specific rules. These rules ensure that the program runs correctly and is readable.

1. First Character Rule

- The first character of a variable must be a letter (A–Z, a–z).
- An underscore _ can also be used at the beginning, but it is not recommended for general use.
- **Reason**: If a number or symbol is used at the start, the compiler will give an error.

Examples:

- **Valid**: age, totalMarks, _count
- **Invalid**: 2kg, #value

2. Characters Allowed

- After the first characteur, a variable name can include:
  - Letters (A–Z, a–z)
  - Digits (0–9)
  - Underscore _
- **Examples**: marks1, student_count, total3Score

3. Case Sensitivity

- C is case-sensitive, meaning uppercase and lowercase letters are treated differently.
- Count ≠ count

Exam Tip: Always be consistent with naming.

4. No Keywords

- Keywords are reserved words in C (like int, while, return) that have a special meaning in programming.
- They cannot be used as variable names.
- Example of invalid variable names: int, float

5. Maximum Length

- Most C compilers allow variable names up to 31 characters, though the name can be longer in some compilers.
- Only the first 31 characters are considered by the compiler.
- **Example:** problemsolvingtechniquesinC (first 31 characters matter)

6. No Spaces

- Variable names cannot contain spaces.
- **Invalid**: total marks
- **Valid**: total_marks

7. Single Data Type

- A variable can store only one type of data (int, float, char, etc.) as declared.
- You cannot assign a string to an integer variable.

8. Readability

- Variable names should be meaningful so anyone reading the program understands its purpose.

- **Example:**
    - **Good**: interestRate, totalStudents
    - **Bad**: x, y, a1

✦ **Summary:**

- Start with a letter (or underscore).
- Can include letters, digits, underscore.
- Case-sensitive, no keywords, no spaces.
- Max 31 characters.
- Should be readable and meaningful.

🌟 **Q.7: Differentiate the following:**

(i) Constant and Variable

- A constant is a value that cannot be changed during the execution of a program. For example, #define PI 3.142 is a constant. Once defined, its value remains fixed throughout the program.
- **A variable** is a memory location whose value can change during program execution. For example, int age = 20; allows the value of age to be updated later in the program. Constants are usually determined at compile-time, while variables are stored in memory at

runtime. Constants are used for fixed values such as π or tax rates, while variables store dynamic data.

(ii) Character Constant and Numeric Constant

- **A character constant** represents a single character enclosed in apostrophes ' '. Examples include 'A', '5', or '%'. Its length is always one character, and it can be a letter, digit, or symbol.
- **A numeric constant** represents a number that can be either an integer or a floating-point value, and its value is fixed in the program. Examples are 56, -4.78, or 0.45. Numeric constants are used in arithmetic operations and calculations.

(iii) Standard Data Type and User-Defined Data Type

- A standard data type is predefined in the C language. Examples include int, float, double, and char. These types are used to store basic forms of data.
- A user-defined data type is created by the programmer according to the program's specific needs. Examples include struct, enum, and typedef. User-defined types allow the creation of custom data structures to handle more complex or specific data.

(iv) Keyword and Identifier

- A keyword is a reserved word in C that has a predefined meaning. Examples are int, return, if, while. Keywords cannot be redefined by the programmer and help the compiler understand the program structure.
- An identifier is a name chosen by the programmer to represent variables, functions, constants, or labels. Examples include age, sum, totalMarks. Identifiers are user-defined and can be used and manipulated in the program's logic.

✦ **Summary:**

- A constant is fixed, while a variable can change.
- Character constants are single characters, numeric constants are numbers.
- Standard data types are predefined, user-defined types are custom.
- Keywords are reserved, identifiers are programmer-defined.

🌟 Q.8: What is a Data Type? Discuss various C data types to manipulate integers, floating point numbers, and characters.

A data type in C is a classification that specifies the type of data a variable can store and the operations that can be performed on that data. It tells the compiler how much memory to allocate for the variable and what kind of values it can hold. In simple words, data types help the program understand the kind of data it is dealing with.

1. Data Types for Integers

Integer data types are used to store whole numbers, i.e., numbers without a fractional part.

- **int:** Standard integer type. Can store positive and negative numbers. For example, int age = 20;. Memory: 2 bytes (range: -32768 to 32767).
- **short int (short):** Used for smaller integers to save memory. Memory: 2 bytes (similar range to int).
- **long int (long):** Used for larger integers. Memory: 4 bytes (range: -2147483648 to 2147483647).

- **unsigned int:** Stores only positive integers, including zero. Cannot store negative numbers. Memory: 2 bytes (range: 0 to 65535).

All integer types can be signed (default) or unsigned.

2. Data Types for Floating Point Numbers

Floating point data types are used to store numbers with fractional parts, e.g., 3.14, -8.75, 0.001.

- **float:** Single precision floating point. Memory: 4 bytes. Example: float price = 45.75;.
- **double:** Double precision, more accurate than float. Memory: 8 bytes. Example: double distance = 123.45678;.
- **long double:** Extended precision for very large or very small numbers. Memory: 12–16 bytes depending on the compiler.

Floating point numbers are stored in scientific notation, with a mantissa (value) and an exponent (power of 10).

3. Data Type for Characters

The char data type is used to store a single character, such as a letter, digit, or symbol.

- **Memory:** 1 byte. Example: char grade = 'A';
- Characters are stored in ASCII code, so operations like addition or comparison are performed on their numeric ASCII values.
- **signed char:** Can store negative values (-128 to 127).
- **unsigned char:** Can store only positive values (0 to 255).

✦ **Summary:**

- Integer types (int, short, long) store whole numbers.
- Floating point types (float, double, long double) store fractional numbers.
- Character type (char) stores single characters.
- Data types determine memory allocation, range of values, and operations allowed on variables.

🌟 **Q.9: How many types of operators are available in C? Describe briefly. Also mention their precedence.**

Operators are special symbols in C that allow us to perform operations on data. They are like tools that help the computer manipulate numbers, variables, or logic conditions.

C has several types of operators, which can be grouped as follows:

1. Arithmetic Operators

These operators are used to perform mathematical calculations.

- **Symbols:** + (addition), - (subtraction), * (multiplication), / (division), % (modulus or remainder)
- **Example:**

C

```
int a = 10, b = 3;

int sum = a + b;  // sum = 13

int remainder = a % b; // remainder = 1
```

- **Note:** The modulus % gives the remainder of an integer division, unlike / which gives the quotient.

2. Relational Operators

Relational operators are used to compare two values. They return true (1) or false (0).

- **Symbols:** == (equal to), != (not equal to), < (less than), > (greater than), <= (less than or equal to), >= (greater than or equal to)

Example:

C

```
if (a > b) {
    printf("a is greater than b");
}
```

These operators are useful in decision-making statements like if or while.

3. Logical Operators

Logical operators are used to combine two or more conditions.

- **Symbols:**
  - && ⟶ logical AND
  - ||⟶ logical OR
  - ! ⟶ logical NOT
  - **Example:**

C

```
int salary = 9000;
char status = 'M';
```

```
if (salary < 10000 && status == 'M') {
    printf("Employee gets relief allowance");
}
```

Explanation:

- && ⟶ true if both conditions are true
- ||⟶ true if any one condition is true
- ! ⟶ reverses the result of a condition

4. Assignment Operator

- The assignment operator is used to store a value in a variable.
- **Symbol**: =
- **Example:**

C

```
int x;
x = 10;  // assigns value 10 to x
```

- The right-hand side is evaluated first, then the value is stored in the left-hand side variable.

5. Increment and Decrement Operators

These operators are used to increase or decrease a variable by 1.

- **Symbols**: ++ (increment), -- (decrement)
- **Example:**

C

int a = 10;

a++; // a becomes 11

++a; // a becomes 12

- **Prefix vs Postfix:**
  - Prefix (++a) ⟶ increases first, then uses the value
  - Postfix (a++) ⟶ uses the value first, then increases

6. Compound Assignment Operators

These operators combine arithmetic operation with assignment.

- **Symbols**: +=, -=, *=, /=
- **Example:**

C

int a = 10;

a += 5; // equivalent to a = a + 5; a = 15

a *= 2; // equivalent to a = a * 2; a = 30

Operator Precedence

Operator precedence determines which operator is executed first in an expression.

From highest to lowest precedence:

1. ! ⟶ logical NOT
2. ++, -- ⟶ increment/decrement
3. *, /, % ⟶ multiplication, division, modulus
4. +, - ⟶ addition, subtraction
5. <, <=, >, >= ⟶ relational operators
6. ==, != ⟶ equality and inequality
7. && ⟶ logical AND
8. || ⟶ logical OR
9. = ⟶ assignment

Rules:

- Operators with higher precedence are evaluated first.
- Operators of the same precedence are evaluated left to right, except for assignment (=) which is right to left.

✦ **Summary:**

Operators are the backbone of programming in C. Without operators, we cannot perform calculations, comparisons, or logical decisions. Understanding types of operators and their precedence is essential for writing correct programs.

🌟 **Q.10: What data type would you use to represent the following items: number of children at your school, a letter grade on an exam, and the average marks of your class?**

❖ **Answer:**

To represent different types of data in C, we select the data type depending on the nature of the value:

1. Number of children at your school

- This is a whole number (0, 1, 2, ...), which does not have a fractional part.
- Data type: int (integer)
- Example in C:

C

```
int numberOfChildren = 500;
```

2. A letter grade on an exam

- This is a single character, such as 'A', 'B', 'C', etc.
- **Data type**: char (character)
- **Example** in C:

C

```
char grade = 'A';
```

3. Average marks of your class

- This is a number that may have a decimal (fractional part), e.g., 78.5.
- **Data type:** float or double (floating point number)
- **Example** in C:

C

```
float averageMarks = 78.5;
```

✦ **Summary:**

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

🌟 Q.11: Which of the following are valid variable names in C?

❖ **Answer:**

In C, variable names must follow these rules:

- They can contain letters, digits, and underscores _.
- They cannot start with a digit.
- Keywords (like double, long) cannot be used as variable names.
- Spaces and special symbols (like #, -) are not allowed.
- C is case-sensitive, so Item and item are different.

Check each name:

(i) income ✅ Valid

(ii) total marks ❌ Invalid (contains space)

(iii) double ❌ Invalid (keyword)

(iv) average-score ❌ Invalid (contains -)

(v) room# ❌ Invalid (contains #)

(vi) _area ✅ Valid (underscore allowed at start)

(vii) no_of students ❌ Invalid (contains space)

(viii) long ❌ Invalid (keyword)

(ix) Item ✅ Valid

(x) MAX_SPEED ✅ Valid

Valid variable names:

income, _area, Item, MAX_SPEED ✅

🌟 Q.12: Write a note on the following:

(i) Arithmetic Exprekssion:

An arithmetic expression in C is a combination of operators and operands that produces a value. Operands can be variables, constants, or numbers, and operators include arithmetic symbols like +, -, *, /, and %.

- For example, a + b, x * y, or 7 + m are arithmetic expressions.
- Arithmetic expressions are used to perform mathematical calculations in a program.
- The data type of an expression depends on the type of its operands. If both operands are integers, the result is an integer. If at least one operand is a floating-point number, the result is a floating-point value.

- Special care must be taken with division. Division of two integers results in truncated integer division, while including a float gives the accurate result.
    - **Example**: 7 / 2 results in 3
    - **Example**: 7.0 / 2 results in 3.5

(ii) Comments in C:

Comments are notes written in the program to explain code, increase readability, and help in debugging. They are ignored by the compiler.

- **Single-line comments:** Start with // and extend to the end of the line.
    - **Example:** // This line prints the sum of two numbers
- **Multi-line comments:** Begin with /* and end with */. They can span multiple lines.
- **Example:**

```
/* This program calculates the
   factorial of a given number */
```

- Comments are very useful for documenting code, making it easier for programmers to understand or modify it later.

🌟 Q.13: Correcting Arithmetic Expressions in C

Question:

Let w, x, y, z be four variables of type float, and a, b, c be three variables of type int. Each of the following statements contains one or more violations of the rules for forming arithmetic expressions in C. Rewrite these statements so that they are consistent with the rules of C.

Statements to correct:

z = 4.0 w^ * y

y = yz

a = 6b4;

c = 3(a + b);

z = 7w + xy;

Solution and Explanation:

(i) Original:

z = 4.0 w^ * y

Errors:

- 4.0 w ⟶ Missing multiplication operator *.
- ^ ⟶ In C, ^ is bitwise XOR, not multiplication. Cannot use it with floats.
- There are extra spaces and symbols that are not valid.

Corrected Statement:

z = 4.0 * w * y;

Explanation:

- The multiplication operator * must be explicitly written between constants and variables.
- Floats can only be multiplied using * operator.
- Always end the statement with a semicolon ;.

(ii) Original:

y = yz

Errors:

yz ⟶ Implied multiplication is not allowed in C. Must explicitly use *.

Corrected Statement:

y = y * z;

Explanation:

- C does not understand concatenated variable names as multiplication.
- Each operand must be separated by an operator.

(iii) Original:

a = 6b4;

Errors:

6b4 ⟶ Numbers and variables cannot be written together without an operator.

Corrected Statement:

a = 6 * b * 4;

Explanation:

- Multiplication must be explicit.
- Multiple constants and variables must each be separated by *.

(iv) Original:

c = 3(a + b);

Errors:

3(a + b) ⟶ Implied multiplication is not allowed. Must use * operator.

Corrected Statement:

c = 3 * (a + b);

Explanation:

- Parentheses ( ) are allowed for grouping.
- Multiplication by a constant must use *.

(v) Original:

z = 7w + xy;

Errors:

- 7w ⟶ Missing multiplication operator *.
- xy ⟶ Missing multiplication operator *.

Corrected Statement:

z = 7 * w + x * y;

Explanation:

- Multiplication must be explicit.

- Operator precedence ensures that multiplication is done first, then addition.
- **Semicolon** ; ends the statement.

Key Points to Remember:

1. No implied multiplication: Always use * operator.
2. **Operands:** Variables and constants must be separated by operators.
3. **Operator precedence:**
    - () ⟶ Parentheses (highest)
    - * and / ⟶ Multiplication and division
    - + and - ⟶ Addition and subtraction
    - = ⟶ Assignment (lowest)

Semicolon ; is mandatory at the end of each statement.

🌟 Q.14: Assume that you have the following variable declarations:

int a = 2, b, c = 1, d = 3, p;

float v, w, x, y = 0.3E+1, z = 1.3;

Evaluate each of the following statements:

v = a^ * 2.5 / y ;

w = a / y;

p = a / d;

x = (a + c) / (z + 0.3);

b = d / a + d % a;

y = c/d^ * a;

Solution with Corrections and Evaluations:

(i) v = a^ * 2.5 / y ;

Correction: v = a * 2.5 / y; (a^ is invalid, use *)

Evaluation:

v = 2 * 2.5 / 3.0

v = 5.0 / 3.0

v ≈ 1.667

✅ v ≈ 1.667

(ii) w = a / y;

Already correct.

Evaluation:

w = 2 / 3.0

w ≈ 0.667

✅ w ≈ 0.667

(iii) p = a / d;

Integer division (both a and d are int) ⟶ fractional part is truncated.

Evaluation:

p = 2 / 3

p = 0

✅ p = 0

(iv) x = (a + c) / (z + 0.3);

- Already correct.
- Evaluation:

a + c = 2 + 1 = 3

z + 0.3 = 1.3 + 0.3 = 1.6

x = 3 / 1.6

x ≈ 1.875

✅ x ≈ 1.875

(v) b = d / a + d % a;

- Already correct.
- Evaluation:

d / a = 3 / 2 = 1 // integer division

d % a = 3 % 2 = 1 // remainder

b = 1 + 1 = 2

✅ b = 2

(vi) y = c/d^ * a;

Correction: y = c / (float)d * a; (d^ invalid, cast to float for correct division)

Evaluation:

y = 1 / 3.0 * 2

y ≈ 0.3333 * 2

y ≈ 0.667

✅ y ≈ 0.667

✦ **Summary Table:**

Variable	Correct Expression	Value
v	v = a * 2.5 / y	1.667
w	w = a / y	0.667
p	p = a / d	0
x	x = (a + c) / (z + 0.3)	1.875
b	b = d / a + d % a	2
y	y = c / (float)d * a	0.667

Note:

This chapter is designed to provide a solid foundation of knowledge, with the goal of deepening understanding and encouraging further exploration of the subject. The content has been carefully selected to support effective learning and inspire students to engage with the topic more deeply.

Author: Muhammad Asghar

Purpose: To contribute to education by offering insightful, valuable content that enhances learning and understanding.

Copyright & Usage Policy