Standard I/O in 'C':

Standard Input/Output (I/O) in C refers to the methods used for taking input from the user and displaying output to the user through the console. In C, this is primarily achieved through the use of functions from the <stdio.h> header file. Some commonly used functions include:

printf(): Used to display output to the console.
scanf(): Used to take input from the user.
getchar() and putchar(): Used for character input and output, respectively.
These functions provide a convenient way to interact with the user and handle input/output operations in C programs.

C Fundamentals:

C is a high-level programming language developed by Dennis Ritchie in the early 1970s. It is widely used for system programming, application programming, and embedded systems development. Some fundamental concepts in C include:

Syntax: C programs consist of functions, variables, statements, and expressions.
Data types: C supports various data types such as int, float, char, etc.
Control structures: C provides constructs like if, else, switch, while, do-while, and for for controlling the flow of execution.
Functions: C allows the creation of functions for code modularity and reusability.
C Character Set, Constants, Variables, Keywords, and Identifiers:
Character Set: C uses the ASCII character set to represent characters.
Constants: Constants are fixed values that cannot be changed during program execution. They can be numeric constants (integer constants, floating-point constants) or character constants.
Variables: Variables are named memory locations used to store data during program execution. They must be declared before use.
Keywords: Keywords are reserved words in C that have predefined meanings. Examples include int, float, if, else, etc.
Identifiers: Identifiers are names given to variables, functions, arrays, etc., by the programmer. They must follow certain rules, such as starting with a letter or underscore and consisting of

letters, digits, or underscores.

Data Types and Declaration:

C supports various data types, including:

Integer types: int, short, long, unsigned int, etc.

Floating-point types: float, double.

Character types: char.

Derived types: Arrays, pointers, structures, unions, enums.

Variables must be declared before they are used, specifying their data type. For example:

c

Copy code

```
int age; // Declaration of an integer variable 'age'
float height; // Declaration of a floating-point variable 'height'
char grade; // Declaration of a character variable 'grade'
```

Operators and Expressions:

C supports various operators for performing operations on operands. These include arithmetic operators (+, -, *, /, %), relational operators (==, !=, <, >, <=, >=), logical operators (&&, ||, !), etc. Expressions are combinations of operators and operands that result in a value.

Conditional Statements and Decision Making:

if Statement: Allows for conditional execution of code based on a specified condition.

if-else Statement: Provides an alternative execution path if the condition evaluates to false.

Nesting of if-else Statements: Allows for the embedding of if-else statements within other if-else statements.

switch Statement: Provides a convenient way to select one of several code blocks to execute based on the value of a variable or expression.

The ?: Operator: Also known as the ternary conditional operator, it provides a concise way to express conditional expressions.

Looping and Control Structures:

while Loop: Executes a block of code repeatedly as long as a specified condition is true.

do-while Loop: Similar to the while loop, but the condition is evaluated after the execution of the loop body, ensuring that the loop body is executed at least once.

for Loop: A compact loop that allows for specifying initialization, condition, and increment/decrement in a single line.

Break and Continue Statements: Used to control the flow of execution within loops. Break terminates the loop, while continue skips the remaining code in the current iteration and proceeds to the next iteration.

Switch Statement: Already discussed under decision making.

C Programs Based on Above Concepts:

These concepts form the building blocks of C programming. By combining these concepts, programmers can write various programs to perform tasks ranging from simple calculations to complex data processing and manipulation.