perplexity

C Fundamentals - Character Set, Constants, Identifiers, Keywords, Basic

Data types, Variables, Operators and its precedence, Bit-wise operators, Expressions; Statements - Input and Output statements; Structure of a C program; Simple programs.
Control Statements - if, if-else, nested if, switch, while, do-while, for, break & continue, nested loops

teach me all this with appropriate points, differences to note, syntax and an example

Absolutely! Here's a concise, structured guide to the **fundamentals of C**, including control statements, operators, program structure, and examples.

1. C Character Set

- **Letters:** A-Z, a-z
- **Digits:** 0-9
- **Special Characters:** +, -, *, /, =, %, &, #, etc.
- **Whitespace:** Space, tab, newline

2. Constants

- **Integer Constants:** `10`, `-25`
- **Floating-point Constants:** `3.14`, `-0.001`
- **Character Constants:** `'A'`, `'9'`
- **String Constants:** `"Hello"`, `"123"`

3. Identifiers

- Names for variables, functions, arrays, etc.
- **Rules:** Start with letter/underscore, followed by letters, digits, underscores. Case-sensitive.
- **Examples:** `sum`, `TotalMarks`, `_temp`

4. Keywords

- Reserved words with special meaning.
- **Examples:** `int`, `float`, `if`, `else`, `for`, `while`, `return`, `break`, `continue`, `switch`, `case`, `default`, `void`, `char`, `double`, `long`, `short`, `unsigned`, `signed`, `struct`, `union`, `typedef`, `enum`, `auto`, `static`, `extern`, `register`, `const`, `volatile`, `sizeof`, `goto`, `do`

5. Basic Data Types

Type	Size (bytes)	Example
`char`	1	`'A'`, `char c = 'B';`
`int`	2 or 4	`int n = 5;`
`float`	4	`float x = 1.23;`
`double`	8	`double y = 3.1415;`

6. Variables

- Named memory locations to store data.
- **Declaration:** `int age; float salary;`
- **Initialization:** `int age = 25;`

7. Operators and Precedence

Arithmetic Operators

`+`, `-`, `*`, `/`, `%`

Relational Operators

`==`, `!=`, `<`, `>`, `<=`, `>=`

Logical Operators

`&&`, `||`, `!`

Assignment Operators

`=`, `+=`, `-=`, `*=`, `/=`, `%=`

Increment/Decrement

`++`, `--`

Bitwise Operators

`&` (AND), `|` (OR), `^` (XOR), `~` (NOT), `<<` (left shift), `>>` (right shift)

Operator Precedence (Highest to Lowest)

1. Parentheses `()`
2. Unary `++`, `--`, `!`, `~`
3. Multiplicative `*`, `/`, `%`
4. Additive `+`, `-`
5. Shift `<<`, `>>`
6. Relational `<`, `<=`, `>`, `>=`
7. Equality `==`, `!=`
8. Bitwise AND `&`
9. Bitwise XOR `^`
10. Bitwise OR `|`
11. Logical AND `&&`
12. Logical OR `||`
13. Conditional `? :`
14. Assignment `=`, `+=`, `-=`, etc.
15. Comma `,`

8. Expressions

- Combination of variables, constants, operators.
- **Example:** `a + b * c - 10`

9. Statements

- **Input:** `scanf("%d", &num);`
- **Output:** `printf("Result is %d", result);`

10. Structure of a C Program

```
#include <stdio.h> // Preprocessor directive

int main() {       // Main function
    // Variable declarations
    // Input statements
    // Processing (expressions/statements)
    // Output statements
    return 0;      // End of program
}
```

11. Simple Program Example

```
#include <stdio.h>
int main() {
    int a, b, sum;
    printf("Enter two numbers: ");
    scanf("%d %d", &a, &b);
    sum = a + b;
    printf("Sum = %d\n", sum);
    return 0;
}
```

12. Control Statements

if

```
if (condition) {
    // code
}
```

Example:

```
if (a > b) printf("a is greater\n");
```

if-else

```
if (condition) {
    // code if true
} else {
    // code if false
}
```

nested if

```
if (cond1) {
    if (cond2) {
        // code
    }
}
```

switch

```
switch (expression) {
    case value1:
        // code
        break;
    case value2:
        // code
        break;
    default:
        // code
}
```

Example:

```
switch (ch) {
    case 'A': printf("Apple\n"); break;
    case 'B': printf("Banana\n"); break;
    default: printf("Other\n");
}
```

while

```
while (condition) {
    // code
}
```

do-while

```
do {
    // code
} while (condition);
```

for

```
for (init; condition; increment) {
    // code
}
```

Example:

```
for (int i = 0; i < 5; i++) {
    printf("%d ", i);
}
```

break & continue

- **break:** Exits the nearest loop or switch.
- **continue:** Skips to the next iteration of the loop.

Example:

```
for (int i = 0; i < 10; i++) {
    if (i == 5) continue;
    if (i == 8) break;
    printf("%d ", i);
}
```

nested loops

```
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 2; j++) {
        printf("%d %d\n", i, j);
    }
}
```

13. Key Points & Differences

- **Keywords** are reserved; cannot be used as identifiers.
- **Operators** follow precedence; use parentheses to control order.
- **Bitwise operators** work on bits, not values.
- **Control statements** direct program flow.
- **Statements** end with a semicolon.
- **Variables** must be declared before use.

14. Summary Table

Concept	Syntax Example	Notes
Variable	`int a = 5;`	Must declare before use
Input	`scanf("%d", &a);`	Use & for address
Output	`printf("a=%d", a);`	%d for int, %f for float
If	`if (a > 0) { ... }`	
For loop	`for (i=0; i<n; i++)`	
While loop	`while (a < 10)`	
Switch	`switch (var) { ... }`	
Break/Continue	`break;`, `continue;`	

This guide gives you a complete overview of C fundamentals and control statements, with syntax, examples, and key points for each. If you want detailed code for any part, just ask!