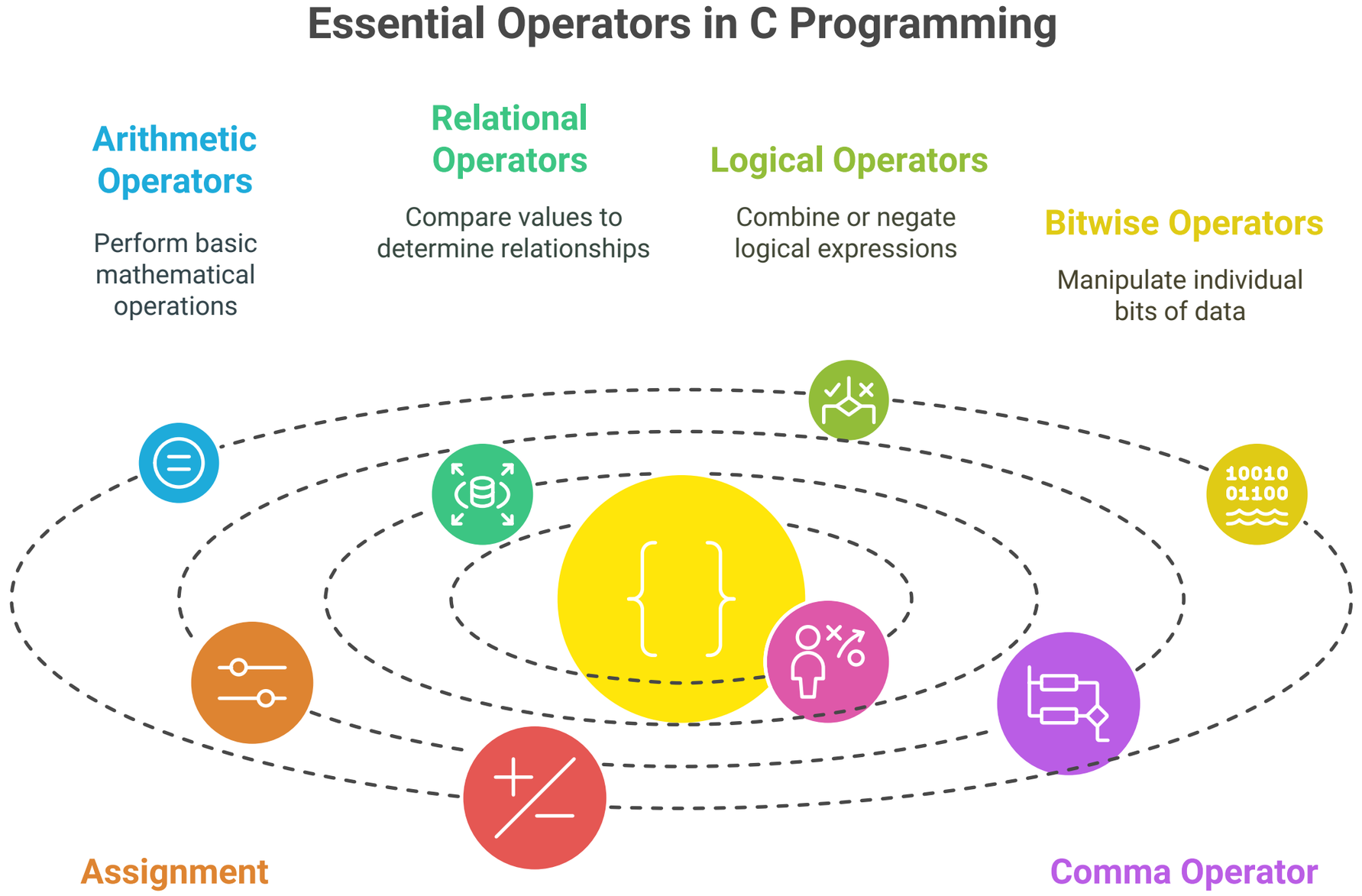
{ } Operators in C Programming

In C programming, operators are special symbols that perform operations on variables and values. They are essential for manipulating data and controlling the flow of execution in a program. This document provides an overview of the various types of operators in C, along with example code and explanations for each type.



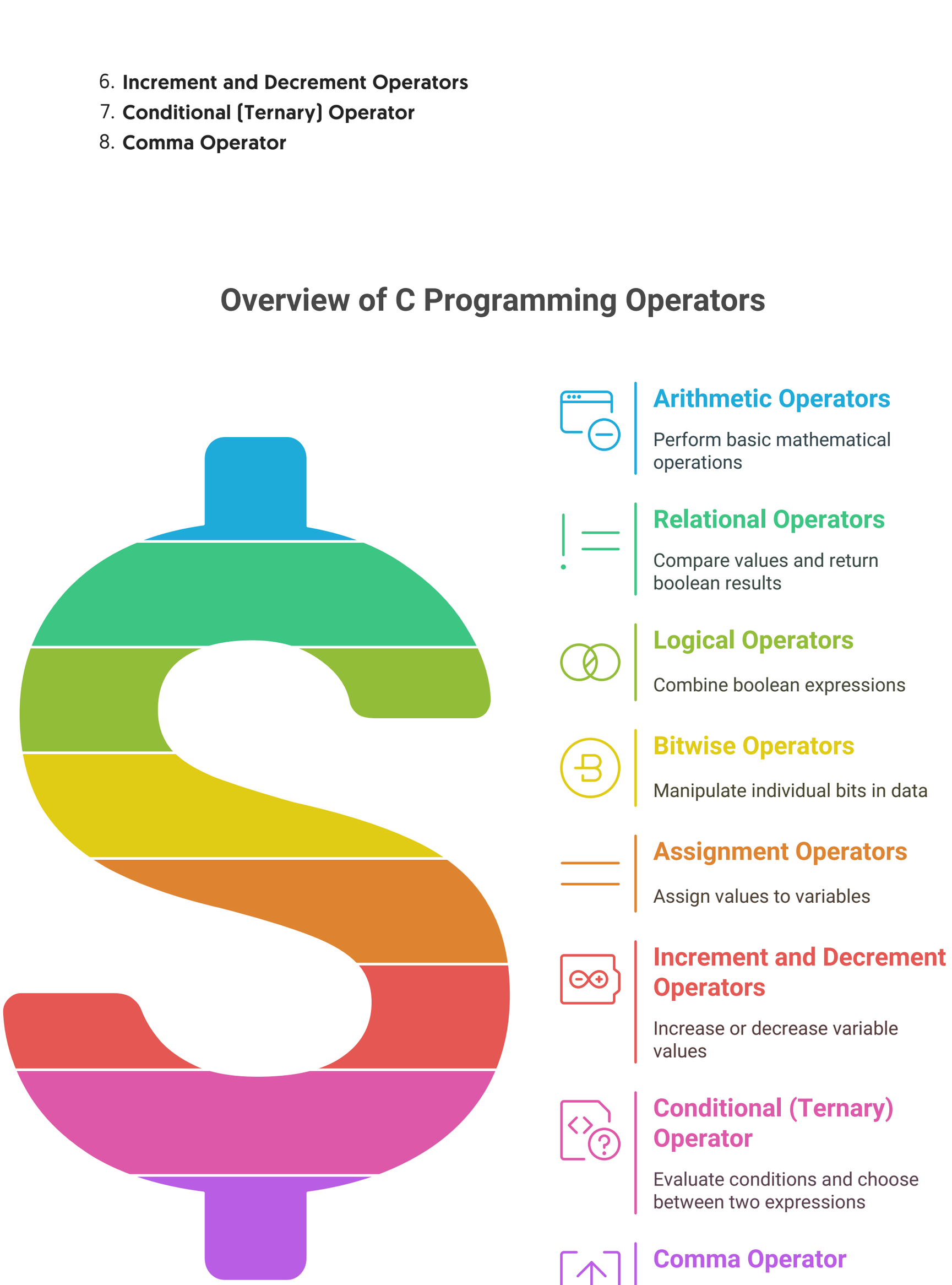
Types of Operators

C programming includes several types of operators, which can be categorized as follows:

1. **Arithmetic Operators**
2. **Relational Operators**
3. **Logical Operators**
4. **Bitwise Operators**
5. **Assignment Operators**

6. **Increment and Decrement Operators**
7. **Conditional (Ternary) Operator**
8. **Comma Operator**

Overview of C Programming Operators



1. Arithmetic Operators

Arithmetic operators are used to perform basic mathematical operations.

Operator	Description	Example
+	Addition	**a + b**
-	Subtraction	**a - b**

*	Multiplication	**a * b**
/	Division	**a / b**
%	Modulus (Remainder)	**a % b**

Example:

```
#include <stdio.h>

int main() {
    int a = 10, b = 3;
    printf("Addition: %d\n", a + b);        // 13
    printf("Subtraction: %d\n", a - b);     // 7
    printf("Multiplication: %d\n", a * b);  // 30
    printf("Division: %d\n", a / b);         // 3
    printf("Modulus: %d\n", a % b);          // 1
    return 0;
}
```

2. Relational Operators

Relational operators are used to compare two values.

Operator	Description	Example
==	Equal to	**a == b**
!=	Not equal to	**a != b**
>	Greater than	**a > b**
<	Less than	**a < b**
>=	Greater than or equal	**a >= b**
<=	Less than or equal	**a <= b**

Example:

```
#include <stdio.h>

int main() {
    int a = 5, b = 10;
    printf("Is a equal to b? %d\n", a == b); // 0 (false)
    printf("Is a not equal to b? %d\n", a != b); // 1 (true)
    printf("Is a greater than b? %d\n", a > b); // 0 (false)
    return 0;
}
```

3. Logical Operators

Logical operators are used to combine multiple conditions.

Operator	Description	Example
&&	Logical AND	**a && b**
||	Logical OR	**a || b**
!	Logical NOT	**!a**

Example:

```
#include <stdio.h>

int main() {
    int a = 1, b = 0;
    printf("Logical AND: %d\n", a && b); // 0 (false)
    printf("Logical OR: %d\n", a || b);  // 1 (true)
    printf("Logical NOT: %d\n", !a);      // 0 (false)
    return 0;
}
```

4. Bitwise Operators

Bitwise operators perform operations on bits and are used for low-level programming.

Operator	Description	Example
&	Bitwise AND	**a & b**
|	Bitwise OR	**a | b**
^	Bitwise XOR	**a ^ b**
~	Bitwise NOT	**~a**
<<	Left shift	**a << 1**
>>	Right shift	**a >> 1**

Example:

```
#include <stdio.h>

int main() {
    int a = 5, b = 3; // 5 = 0101, 3 = 0011
    printf("Bitwise AND: %d\n", a & b); // 1 (0001)
    printf("Bitwise OR: %d\n", a | b);  // 7 (0111)
    printf("Bitwise XOR: %d\n", a ^ b); // 6 (0110)
    return 0;
}
```

5. Assignment Operators

Assignment operators are used to assign values to variables.

Operator	Description	Example
=	Simple assignment	**a = b**
+=	Add and assign	**a += b**
-=	Subtract and assign	**a -= b**
***=**	Multiply and assign	**a *= b**
/=	Divide and assign	**a /= b**
%=	Modulus and assign	**a %= b**

Example:

```c
#include <stdio.h>

int main() {
    int a = 5;
    a += 3; // a = a + 3
    printf("After += : %d\n", a); // 8
    a *= 2; // a = a * 2
    printf("After *= : %d\n", a); // 16
    return 0;
}
```

6. Increment and Decrement Operators

These operators are used to increase or decrease the value of a variable by one.

Operator	Description	Example
++	Increment	**++a** or **a++**
--	Decrement	**--a** or **a--**

Example:

```c
#include <stdio.h>

int main() {
    int a = 5;
    printf("Increment: %d\n", ++a); // 6 (pre-increment)
    printf("Decrement: %d\n", --a); // 5 (pre-decrement)
    return 0;
}
```

7. Conditional (Ternary) Operator

The conditional operator is a shorthand for the if-else statement.

Operator	Description	Example
? :	Ternary operator	**condition ? expr1 : expr2**

Example:

```c
#include <stdio.h>

int main() {
    int a = 5, b = 10;
    int max = (a > b) ? a : b;
    printf("Maximum: %d\n", max); // 10
    return 0;
}
```

8. Comma Operator

The comma operator allows two expressions to be evaluated in sequence.

Example:

```c
#include <stdio.h>

int main() {
    int a, b;
    a = (1, 2, 3); // a will be assigned the value of 3
    printf("Value of a: %d\n", a); // 3
    return 0;
}
```

Conclusion

Operators in C programming are fundamental tools that allow developers to perform a wide range of operations on data. Understanding these operators is crucial for writing effective and efficient C programs. This document has provided an overview of the most commonly used operators, along with examples to illustrate their usage.