Variables and Operators

1 How a Computer is Organized

A computer is made up of several key components that work together to execute programs like your C++ code.

Main Components

Component	Description
CPU	The Central Processing Unit is the brain of the computer. It performs calculations and executes instructions.
RAM	Random Access Memory is short-term memory. It temporarily holds data and instructions that are in use.
Storage	Long-term memory (e.g., SSD or HDD). It permanently stores files, programs, and the operating system.
Input Devices	Devices like the keyboard, mouse, or microphone that allow users to send data into the computer.
Output Devices	Devices such as monitors, speakers, or printers that present data from the computer to the user.

Basic Program Flow

- You write C++ code as high-level instructions.
- The compiler converts this code into machine code.
- The CPU executes the machine code, using RAM to store data temporarily.
- Input/output devices allow the user to interact with the program.

2 What Are Variables in C++?

A **variable** is a named location in memory that stores data.

Common Data Types

Each variable must have a "type" that specifies the type of data it holds. For example, int holds integer data and double holds real numbers.

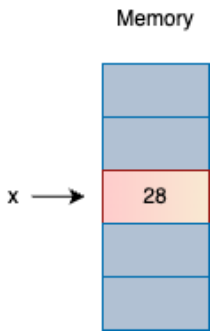

Figure 1: Illustration of variables in memory. Here x is a variable with value 28.

Type	Meaning	Example
`int`	Integer	`int age = 25;`
`float`	Decimal number	`float temp = 98.6;`
`double`	More precise decimal	`double pi = 3.14159;`
`char`	Single character	`char grade = 'A';`
`bool`	True or false	`bool passed = true;`

Constant Variables with `const`

Sometimes you may want to declare a variable whose value should not change after initialization. In such cases, use the `const` keyword. This makes the variable read-only.

```
const double pi = 3.14159;
```

Attempting to modify a `const` variable later in the program will result in a compilation error. This is useful for defining fixed values like mathematical constants, configuration settings, or immutable identifiers.

Note: You must initialize a `const` variable at the time of declaration.

Rules for Naming Variables

- Must start with a letter or underscore (_)
- Cannot use C++ keywords (e.g., `int`, `while`)
- Case-sensitive (`score` and `Score` are different)

3 What Are Operators?

Operators are symbols that tell the computer to perform specific operations on **variables** and **values**.

Operators in C++

Below is a summary of common types of operators in C++:

Arithmetic Operators

Operator	Use	Example	Result
`+`	Addition	`5 + 3`	`8`
`-`	Subtraction	`5 - 3`	`2`
`*`	Multiplication	`5 * 3`	`15`
`/`	Division	`5 / 2`	`2` (integer division)
`%`	Modulus (remainder)	`5 % 2`	`1`

Assignment Operators

Operator	Use	Example
`=`	Assign value	`x = 10;`
`+=`	Add and assign	`x += 5;` (`x = x + 5`)
`-=`	Subtract and assign	`x -= 2;`
`*=`	Multiply and assign	`x *= 3;`

Comparison Operators

Operator	Meaning	Example
`==`	Equal to	`x == 5`
`!=`	Not equal to	`x != 5`
`<`, `>`, `<=`, `>=`	Less than, etc.	`x < 10`

Logical Operators

Operator	Meaning	Example
`&&`	AND (both conditions true)	`x > 0 && y > 0`
`||`	OR (at least one true)	`x > 0 || y > 0`
`!`	NOT (negation)	`!is_valid`

4 How Operators and Variables Work Together

Variables store values. Operators act on these values to compute new results, which can then be stored back in variables.

```
int a = 5;
int b = 2;
int result = a + b;  // '+' is the operator acting on variables a and b
```

In this case:

- `a` and `b` are **variables**
- `+` is an **arithmetic operator**
- The result (7) is stored in another variable `result`

5 Analogy Table

Concept	Role in C++	Analogy
Variable	A box with a name that stores data	A labeled jar
Value	The contents inside the variable	Candy inside the jar
Operator	A tool used to work with values	A spoon to add/subtract candy

6 Operator Actions on Variables

Expression	Action Performed
`x = 10`	Store 10 in variable `x`
`y = x + 5`	Add `x` and 5, store in `y`
`x += 3`	Increase `x` by 3
`z = x * y`	Multiply `x` and `y`, store in `z`

In short: Operators manipulate the values stored in variables to produce new results.

```cpp
#include <iostream>
using namespace std;

int main() {
    // Variable declarations
    int a = 10;
    int b = 3;

    // Arithmetic operators
    int sum = a + b;
    int difference = a - b;
    int product = a * b;
    int quotient = a / b;
    int remainder = a % b;

    // Output results
    cout << "a = " << a << ", b = " << b << endl;
    cout << "Sum (a + b) = " << sum << endl;
    cout << "Difference (a - b) = " << difference << endl;
    cout << "Product (a * b) = " << product << endl;
    cout << "Quotient (a / b) = " << quotient << endl;
    cout << "Remainder (a % b) = " << remainder << endl;

    // Relational operator
    cout << "Is a greater than b? " << (a > b) << endl;

    // Logical operator
    bool result = (a > 5) && (b < 5);
    cout << "Is a > 5 AND b < 5? " << result << endl;

    return 0;
}
```

Comple and run using -

```
g++ demo.cpp -o demo
./demo
```