كليـــة العلـــــوم

قسم الأمن السيبراني

Subject: Programming Fundamentals

First Stage

Assistant Lecturer: Jaber Baqer Al-Hamdani

Lecture (4)

C++ Essentials: Statements, Variables, Constants, and Expression Evaluation

1. Introduction to Statements

Definition: In C++, a statement is an instruction that the compiler can execute. Statements form the basic building blocks of a C++ program.

Types of Statements:

- **Expression Statements:** Perform operations, e.g., `x = y + z;`
- **Compound Statements:** Group multiple statements using `{}`.
- **Control Statements:** Guide program flow, e.g., loops (`for`, `while`) and conditionals (`if`, `switch`).
- **Declaration Statements:** Introduce variables or constants, e.g., `int a;`.

Examples:

```
int x = 10;         // Declaration statement
x = x + 5;          // Expression statement
if (x > 10) {       // Control statement
    cout << x;      // Expression statement within a compound statement
}
```

Key Points:

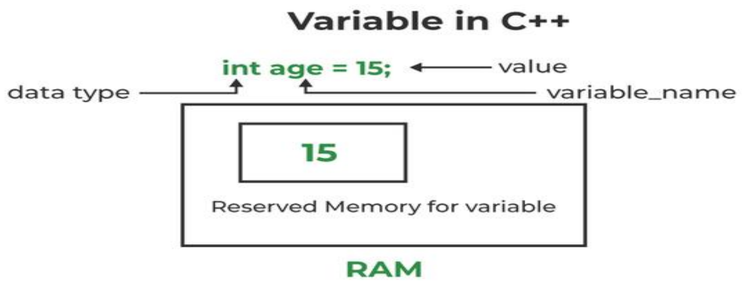
- Each statement ends with a semicolon (`;`).
- Compound statements do not require a semicolon after the closing brace (`}`).

2. Variable Declaration

Definition: A variable is a named location in memory used to store data. Declaring a variable informs the compiler about its name and data type.

Syntax:

```
<data_type> <variable_name> [= initial_value];
```



Examples:

```
int age = 25;                    // Integer variable
float height = 5.9;              // Floating-point variable
char grade = 'A';                // Character variable
string name = "Ali Mohammed";    // String variable
```

Common Data Types:

- `int`: Integer values (e.g., 1, 42, -10)
- `float`: Decimal numbers (e.g., 3.14, -0.01)
- `double`: More precise decimal numbers
- `char`: Single characters (e.g., 'A', 'b')
- `bool`: Boolean values (`true` or `false`)
- `string`: Sequence of characters (e.g., "Hello, World!")

Rules for Naming Variables:

1. Names can include letters, digits, and underscores but cannot start with a digit.
2. Avoid C++ keywords (e.g., `int`, `return`).
3. Use descriptive names for clarity.

3. Constants

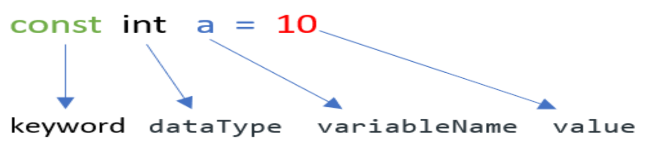
Definition: A constant is a value that does not change during the execution of a program.

Types of Constants in C++:

- **Literal Constants:** Fixed values, e.g., `42`, `3.14`, `'A'`, `"Hello"`.
- **Symbolic Constants:** Declared using the `const` keyword .

Examples:

```
const double PI = 3.14159;      // Constant variable
```



Key Points:

1. Use `const` for variables whose values should not change.

4. Order of Evaluation

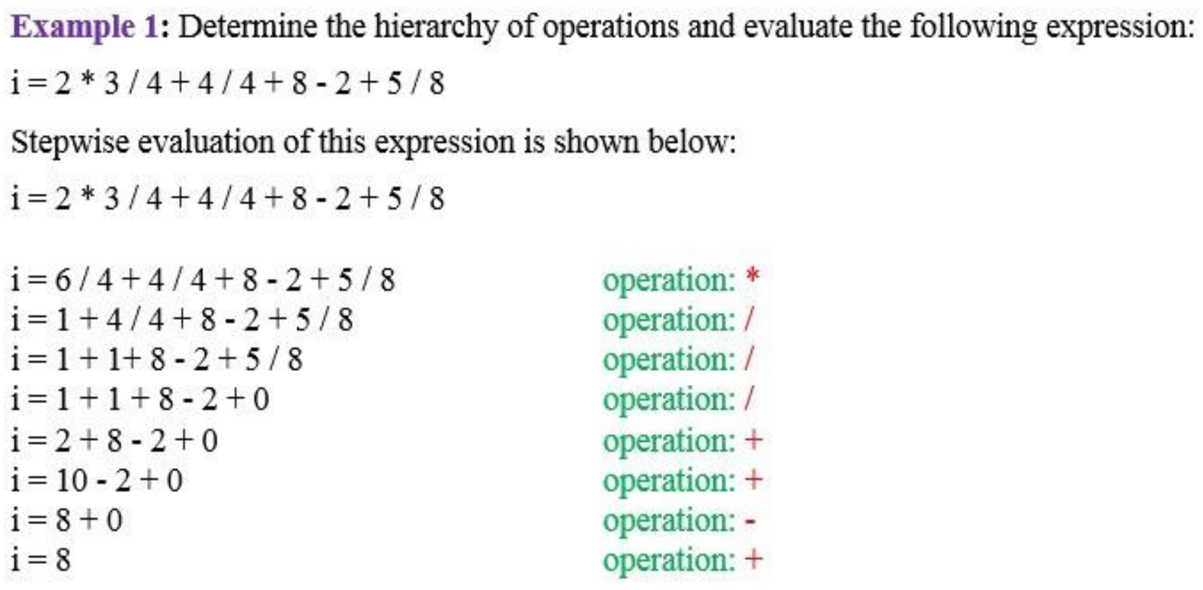
Definition: The order in which expressions are evaluated during execution, especially in compound expressions.

Operator Precedence: Operators in C++ have a defined order of precedence(الاسبقية). For example:

1. **Highest Precedence:** Parentheses `()`
2. Multiplication `*`, Division `/`, Modulus `%`
3. Addition `+`, Subtraction `-`
4. Relational operators (<, >, <=, >=)
5. Equality operators (==, `!=`)
6. Logical AND (`&&`) and Logical OR (`||`)
7. **Lowest Precedence:** Assignment `=`

Associativity: When operators have the same precedence, their associativity(الترابط) determines the evaluation order:

- **Left to Right:** Most operators (e.g., `+`, `-`, `*`, `/`, `%`, <, >)
- **Right to Left:** Assignment (`=`) and some unary operators.



Example 1: Determine the hierarchy of operations and evaluate the following expression:

i = 2 * 3 / 4 + 4 / 4 + 8 - 2 + 5 / 8

Stepwise evaluation of this expression is shown below:

i = 2 * 3 / 4 + 4 / 4 + 8 - 2 + 5 / 8

```
i = 6 / 4 + 4 / 4 + 8 - 2 + 5 / 8        operation: *
i = 1 + 4 / 4 + 8 - 2 + 5 / 8            operation: /
i = 1 + 1+ 8 - 2 + 5 / 8                 operation: /
i = 1 + 1 + 8 - 2 + 0                    operation: /
i = 2 + 8 - 2 + 0                        operation: +
i = 10 - 2 + 0                           operation: +
i = 8 + 0                                operation: -
i = 8                                    operation: +
```

Example 2:

```
int x = 5, y = 10, z;
z = x + y * 2;        // y * 2 is evaluated first, then x + result
```

```
cout << z;              // Outputs: 25

z = (x + y) * 2;        // Parentheses change the order
cout << z;              // Outputs: 30

bool result = (x > 3) && (y < 20);      // Relational operators evaluated
first
cout << result;                         // Outputs: 1 (true)
```

Guidelines:

1. Use parentheses (اقواس) to clarify expressions and avoid ambiguity(الغموض).
2. Be mindful(منتبه او واعي) of the order when combining multiple operators.
3. Test complex expressions step by step to ensure correctness.

Summary

1. **Statements** are the core instructions in C++.
2. **Variables** are declared to store data, with a type defining their behavior.
3. **Constants** ensure data integrity by preventing modification.
4. **Order of Evaluation** influences how expressions are computed and can be controlled using parentheses.

Practice Questions

1. Write a program to declare and initialize variables of different data types.
2. Use `const` to define a constant for the value of PI and calculate the area of a circle.
3. Evaluate the result of the expression `5 + 3 * 2` and modify it to compute `(5 + 3) * 2`.

Assignment

Write a C++ program that:

1. Declares variables for storing your name, age, and GPA.
2. Uses constants to define the maximum possible GPA.
3. Calculates the percentage of your GPA out of the maximum using the correct order of evaluation.