C++ 101 – Session 1

1. Syntax of a C++ Program

📌 Structure Overview:

```
#include <iostream>   // Allows input/output
using namespace std;  // Avoids typing std:: repeatedly

int main() {
    cout << "Hello, World!" << endl; // Output to screen
    return 0;          // Indicates that the program ended successfully
}
```

💡 Explanation:

- Every C++ program must have a `main()` function.
- `cout` is used to display output.
- `endl` moves the cursor to a new line and flushes the output buffer.

---

2. Data Types in C++

Data types tell the compiler what kind of data a variable will store.

Type	Purpose	Example
`int`	Stores whole numbers	`int age = 20;`
`float`	Stores decimal numbers (less precise)	`float pi = 3.14;`
`double`	More precise decimals	`double weight = 45.12;`
`char`	Stores single characters	`char grade = 'A';`
`bool`	Stores true/false values	`bool passed = true;`
`string`	Stores a sequence of characters	`string name = "Ngambo";`

📝 Note: `string` requires including the `#include <string>` header if not already included.

3. Variable Declaration

To use a variable in C++, you must first declare it:

```
int age;
float temperature;
char grade;
```

📌 This tells the compiler:

- What type of data the variable will hold.
- The name you'll use to refer to it.

---

4. Variable Assignment

Once declared, you can assign a value to a variable:

```
age = 25;
temperature = 36.5;
grade = 'B';
```

📝 You can also **declare and assign** in one line:

```
int age = 25;
```

---

5. Comments in C++

Comments are notes you write in your code to explain what it does. They are **ignored by the compiler**.

```
// This is a single-line comment

/* This is a
   multi-line comment */
```

💡 Use comments to make your code easier to understand for yourself and others.

6. Expressions

Expressions perform operations on variables and values.

🔷 6.1. Arithmetic Expressions

Used for mathematical operations:

Operator	Description	Example
`+`	Addition	`a + b`
`-`	Subtraction	`x - y`
`*`	Multiplication	`p * q`
`/`	Division	`a / b`
`%`	Modulus (remainder)	`x % y`

🔷 6.2. Comparison (Relational) Expressions

Used to compare values. They return `true` or `false`.

Operator	Meaning	Example
`==`	Equal to	`x == y`
`!=`	Not equal to	`x != y`
`>`	Greater than	`a > b`
`<`	Less than	`a < b`
`>=`	Greater or equal	`x >= 5`
`<=`	Less or equal	`x <= 10`

📌 These are often used inside `if` or `while` statements.

🔷 3. Logical Expressions

Combine or modify boolean values.

Operator	Name	Example	Result
`&&`	AND	`x > 0 && y > 0`	true if both are true
||	OR	Age > 18 || name = ‘John’	OR
`!`	NOT	`!isCorrect`	true if `isCorrect` is false

🧠 Used in conditions to add more logic.

7. Conditional Statements

Used to make **decisions** in your program based on conditions.

🔹 `if` Statement

Executes code **only if** the condition is true.

```
if (age >= 18) {
    cout << "You are an adult." << endl;
}
```

🔹 `if-else` Statement

Chooses **between two paths**: one if true, another if false.

```
if (score >= 50) {
    cout << "Passed!" << endl;
} else {
    cout << "Failed." << endl;
}
```

🔹 `if-else if-else` (Nested Conditions)

Used to test **multiple conditions** in sequence.

```
if (marks >= 80) {
    cout << "Grade: A" << endl;
} else if (marks >= 60) {
    cout << "Grade: B" << endl;
} else if (marks >= 40) {
    cout << "Grade: C" << endl;
} else {
    cout << "Fail" << endl;
}
```

🧠 The program checks from top to bottom and runs the first condition that is true.

Compiling and Running a C++ Program

If you're using a terminal or command prompt and have a C++ compiler like `g++` installed, follow these steps:

1. **Save your C++ file** with a `.cpp` extension.
   Example: `main.cpp`
2. **Open terminal or command prompt** in the directory where the file is saved.
3. **Compile the file** using `g++`:

   ```
   g++ main.cpp -o main
   ```

4. **Run the executable**:

   ```
   ./program
   ```