C++ 101 – Session 5 Notes

1. Printing a Matrix with Nested `for` Loops

In programming, a **matrix** (or grid) is a collection of values arranged in rows and columns — just like a table.

To print a matrix in C++, we use **nested loops**:

- The **outer loop** handles the rows.
- The **inner loop** handles the columns.

📌 Example:

```
int rows = 5;
int cols = 3;

for(int i = 1; i <= rows; i++) {
    for (int j = 1; j <= cols; j++) {
        cout << "#" << "*" << " ";
    }
    cout << endl;  // moves to the next line after each row
}
```

📜 Output:

```
#* #* #*
#* #* #*
#* #* #*
#* #* #*
#* #* #*
```

🧠 Why this matters:

- This logic is the foundation for working with **2D arrays**, **drawing patterns**, and even **basic game grids**.

---

2. The `break` and `continue` Keywords

🔷 `break` — Exit Early

The `break` statement **immediately exits** the current loop. It's useful when you've found what you're looking for or want to stop due to a certain condition.

```
for (int i = 0; i < 10; i++) {
    if (i == 5) {
        break;  // loop stops running when i reaches 5
    }
    cout << i << endl;
}
```

🔍 Output:

```
0
1
2
3
4
```

🔷 `continue` — Skip This One

The `continue` statement **skips the rest** of the current loop iteration and jumps to the next one. Useful for ignoring specific values without stopping the whole loop.

```
for (int i = 0; i < 10; i++) {
    if (i % 2 == 0) {
        continue;  // skip even numbers
    }
    cout << i << endl;
}
```

🔍 Output:

```
1
3
5
7
9
```

---

3. Arrays in C++

An **array** is a fixed-size container used to store multiple values of the same type.
It's great when you want to group similar data together.

🔷 String Array Example:

```
string cars[6] = {"Volvo", "BMW", "Ford", "Mazda", "Toyota", "Honda"};
```

```
for (string car : cars) {
    cout << car << endl;
}
```

This uses a **range-based for loop** (also called a for-each loop) to print each car.

🔷 Integer Array Example:

```
int age[5] = {25, 17, 10, 15, 19};

for (int i = 0; i < 5; i++) {
    cout << age[i] << endl;
}
```

📌 Note: Arrays are **zero-indexed**, meaning `age[0]` is the first element.

---

4. Random Number Generation in C++

In games, simulations, or anything unpredictable — we need **random numbers**.

C++ provides this using:

- `rand()` – generates a random number
- `srand()` – seeds the random number generator so the results are different every time
- `time(0)` – current time, used as a unique seed

📌 Example:

```
#include <iostream>
#include <cstdlib>  // for rand, srand
#include <ctime>    // for time

using namespace std;

int main() {
    srand(time(0));  // Seed the generator

    int randomNumber = rand() % 100 + 1;  // Random number between 1 and 100
    cout << "Random number: " << randomNumber << endl;

    return 0;
}
```

🔍 Explanation:

- `% 100 + 1` limits the result to 1–100.
- If you don’t use `srand()`, you’ll get the same "random" number every time.

🛠️ Task

Each of you will work on the following:

✅ 1. Random Number Generator

- Write your own program using `rand()` and `srand(time(0))`.
- Explain how it works and how to customize the range.

✅ 2. Multidimensional Arrays

- Declare and initialize a 2D array (like a grid or matrix).
- Print it using **nested loops**.
- Try using real values (e.g., student marks, game boards, etc.)

💬 Notes:

- Submit runnable `.cpp` files.
- Add comments in your code to explain what each part does.
- Be ready to present and explain in the next session.