

Introduction to Programming

Lecture 5: **C++ Flow Control (Repetition)**

Part2: Examples

Dr. Nizar Saadi Dahir

Finding Min and Max in a Stream

- ***The Problem:*** *The user enters a list of numbers (unknown length). The program must find the highest and lowest without storing them in an array.*
- *This is how real-time sensors monitor "Peak" and "Low" values without filling up memory.*

- # Finding Min and Max in a Stream
- ***The Problem:*** *The user enters a list of numbers (unknown length). The program must find the highest and lowest without storing them in an array.*
 - *This is how real-time sensors monitor "Peak" and "Low" values without filling up memory.*

Finding Min and Max in a Stream

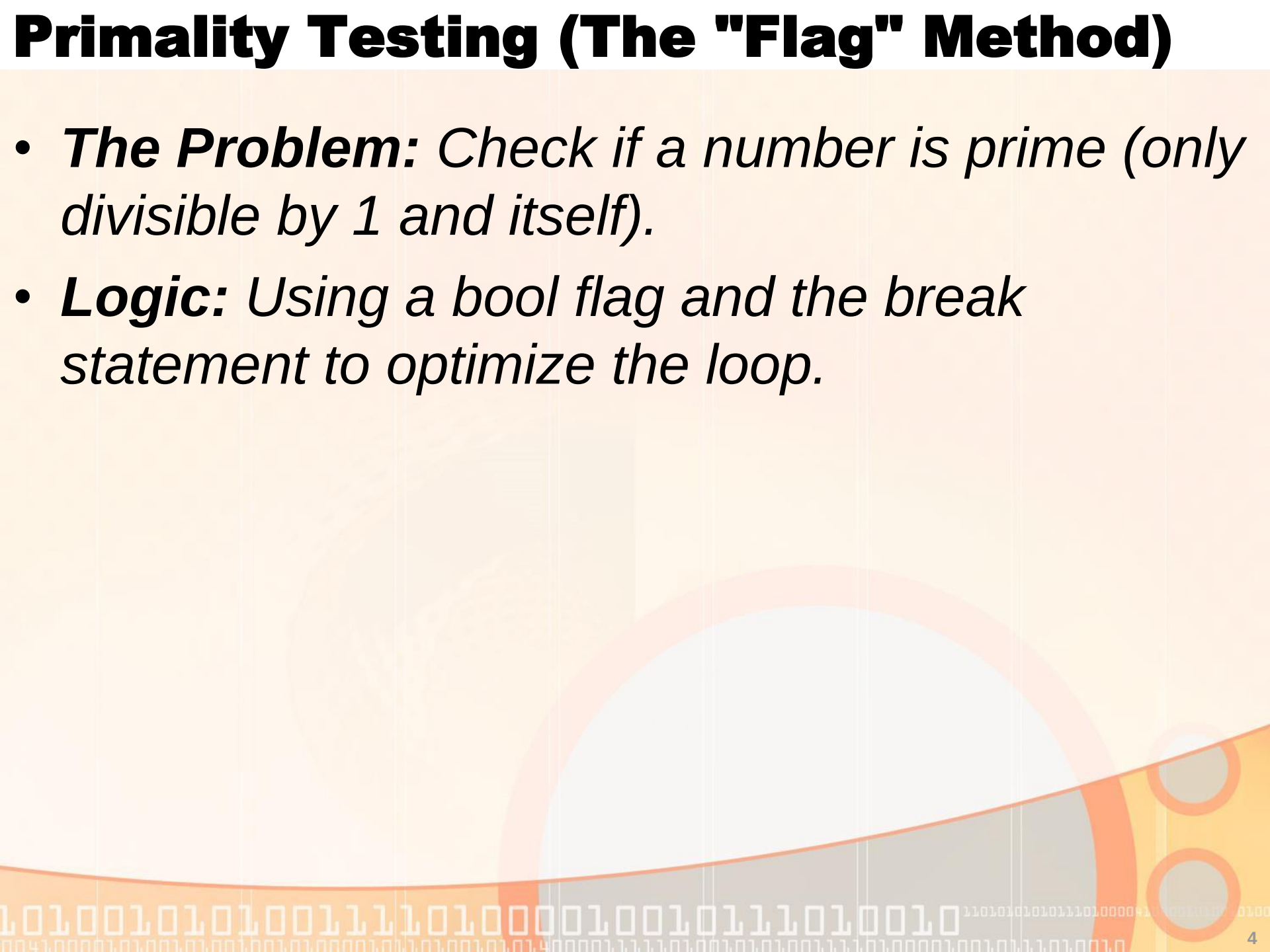
```
int num, max_val, min_val;
cout << "Enter a number (0 to stop): ";
cin >> num;

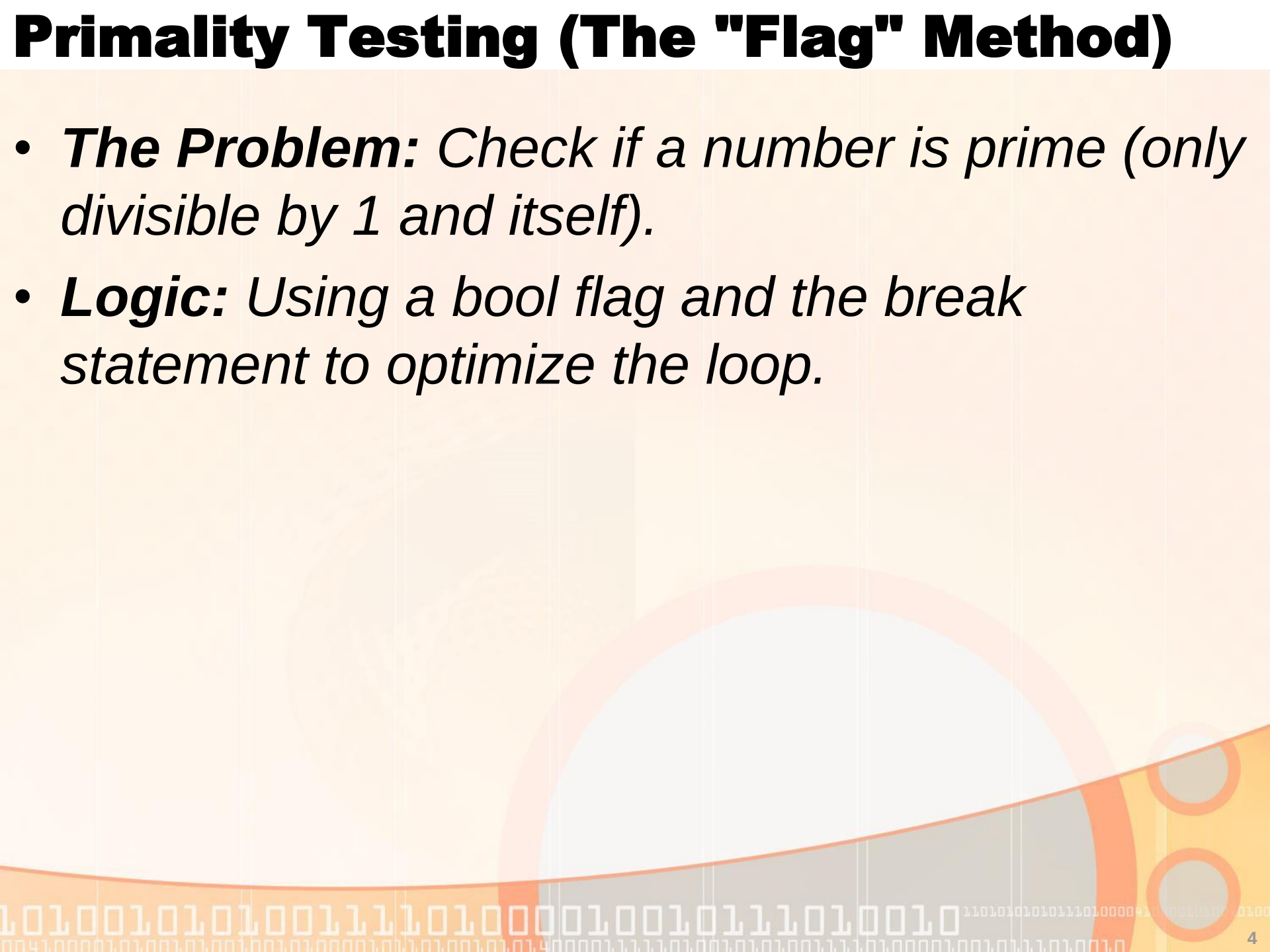
max_val = min_val = num; // Initialize both with the first number

while (num != 0) {
    if (num > max_val) max_val = num;
    if (num < min_val && num != 0) min_val = num;
    cin >> num;
}
cout << "Max: " << max_val << " | Min: " << min_val;
```

Primality Testing (The "Flag" Method)

- ***The Problem:*** Check if a number is prime (only divisible by 1 and itself).
- ***Logic:*** Using a bool flag and the break statement to optimize the loop.



- # Primality Testing (The "Flag" Method)
- ***The Problem:*** Check if a number is prime (only divisible by 1 and itself).
 - ***Logic:*** Using a bool flag and the break statement to optimize the loop.
- 

Primality Testing (The "Flag" Method)

```
int num;
bool is_prime = true;
cout << "Enter a number: ";
cin >> num;

if (num < 2) is_prime = false;
for (int i = 2; i <= num / 2; i++) {
    if (num % i == 0) {
        is_prime = false; // Factor found!
        break;           // Exit loop early for efficiency
    }
}

if (is_prime) cout << num << " is a prime number.";
else cout << num << " is a composite number.";
```

Character Permutations

- **The Goal:** Find every possible 3-letter combination of 'A', 'B', and 'C' without repeating the same letter in a single set (e.g., "ABC" is okay, but "AAA" or "AAB" are not).
- **The Logic:** We use three nested loops. Each loop represents one position in the 3-letter word. Inside the deepest loop, we use an if statement to ensure all three letters are unique.

- # Character Permutations
- ***The Goal:*** Find every possible 3-letter combination of 'A', 'B', and 'C' without repeating the same letter in a single set (e.g., "ABC" is okay, but "AAA" or "AAB" are not).
 - ***The Logic:*** We use three nested loops. Each loop represents one position in the 3-letter word. Inside the deepest loop, we use an if statement to ensure all three letters are unique.

Character Permutations

```
cout << "All unique permutations of A, B, and C:" << endl;

for (char i = 'A'; i <= 'C'; i++) {           // First Position
    for (char j = 'A'; j <= 'C'; j++) {       // Second Position
        for (char k = 'A'; k <= 'C'; k++) {   // Third Position

            // Logic: Only print if all three characters are different
            if (!(i == j || i == k || j == k)) {
                cout << i << j << k << endl;
            }
        }
    }
}
```

Validating User Input (do-while)

- *The Problem: Forcing a user to enter a grade between 0 and 100.*

```
int grade;
do {
    cout << "Enter a valid grade (0-100): ";
    cin >> grade;

    if (grade < 0 || grade > 100) {
        cout << "Error! Invalid range. ";
    }
} while (grade < 0 || grade > 100);

cout << "Grade accepted: " << grade;
```

Decimal to Binary Conversion

- **The Modulo Operator (%):** Used to find the remainder (the binary digit, 0 or 1).
- **The Division Operator (/):** Used to reduce the decimal number for the next iteration.
- **Positional Math:** We multiply the remainder by powers of 10 (1, 10, 100, ... 1 comma 10 comma 100 comma ... 1, 10, 100, ...) to build the final number as a long long..

Decimal to Binary Conversion

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

int main() {
    int decimal;
    long long binary = 0;
    int place = 1;

    cout << "Enter decimal: ";
    cin >> decimal;

    while (decimal > 0) {
        int rem = decimal % 2;
        binary += (rem * place);
        decimal /= 2;
        place *= 10;
    }

    cout << "Binary equivalent: " << binary << endl;
    return 0;
}
```

Decimal to Binary Conversion

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

int main() {
    // Variable Declarations
    int decimal;           // To store the user's input decimal number
    long long binary = 0;  // To accumulate the binary result (long long prevents early overflow)
    int place = 1;         // To track the decimal place (1, 10, 100, 1000, etc.)

    // User Input
    cout << "Enter decimal: ";
    cin >> decimal;

    // Handle the special case where input is 0
    if (decimal == 0) {
        cout << "Binary equivalent: 0" << endl;
        return 0;
    }

    // Conversion Loop: Process the number until it reaches 0
    while (decimal > 0) {
        // Step 1: Find the remainder when divided by 2 (either 0 or 1)
        int rem = decimal % 2;

        // Step 2: Append the remainder to our binary variable at the correct position
        // Example: If binary is 1 and rem is 0 at place 10, binary becomes 01
        binary += (rem * place);

        // Step 3: Divide the decimal number by 2 to process the next bit
        decimal /= 2;

        // Step 4: Shift the place value to the left (next power of 10)
        place *= 10;
    }

    cout << "Binary equivalent: " << binary << endl; // Final Output
    return 0;
}
```