|  |  |
| --- | --- |
| **Leading Zeroes in C++** | |
| #include <iostream>  using namespace std;  int countLeadingZeros(int num) {  int leadingZeros = 0;  while ((num & (1 << 31)) == 0) {  leadingZeros++;  num <<= 1;  }  return leadingZeros;  }  int main() {  int num = 7; // Binary: 00000111  int leadingZeros = countLeadingZeros(num);  cout << "Leading zeros: " << leadingZeros << endl; // Output: 28  return 0;  } | Objective: Count **leading zeros** in the 32-bit binary form of num. 🔢 Step-by-step: The number 7 in **binary (32-bit)** is:  00000000 00000000 00000000 00000111  That's **3 bits set** on the right side — so we expect **29 leading zeros** before the first 1.  Let’s walk through it more carefully. ⚙️ First thing to note:  * 1 << 31 results in a mask:   10000000 00000000 00000000 00000000   * So the code is checking: “Is the **leftmost (31st)** bit in num set?”  🧮 Loop Simulation: Each time, we:   * Check MSB (bit 31) * If zero, we increment leadingZeros * Then do num <<= 1 (left shift)   Let’s track just leadingZeros:   | **Iteration** | **num (binary)** | **MSB** | **leadingZeros** | | --- | --- | --- | --- | | 0 | 00000000 00000000 00000000 00000111 | 0 | 0 | | 1 | 00000000 00000000 00000000 00001110 | 0 | 1 | | 2 | 00000000 00000000 00000000 00011100 | 0 | 2 | | 3 | 00000000 00000000 00000000 00111000 | 0 | 3 | | ... | ... | ... | ... | | 28 | 01000000 00000000 00000000 00000000 | 0 | 28 | | 29 | 10000000 00000000 00000000 00000000 | 1 | 29 (exit loop) |   ✅ So yes — the **loop runs 29 times**, because the **first 1 in the 32-bit form** appears at **bit position 2 (from right)**, i.e., bit index **29 (from left)**. ✅ Final Answer: Leading zeros: 29 |
| Leading zeros: 29 | |