|  |  |
| --- | --- |
| **Reverse bits in C++** | |
| #include <iostream>  using namespace std;  int reverseBits(int num) {  int reversed = 0;  for (int i = 0; i < 32; i++) {  reversed = (reversed << 1) | ((num >> i) & 1);  }  return reversed;  }  int main() {  int num = 25; // Binary: 00011001  int reversed = reverseBits(num);    cout << reversed << endl; // Output: 147  return 0;  } | Input: int num = 25; Binary of 25 (8-bit view): 00000000 00000000 00000000 00011001  Which is: (1\*16) + (1\*8) + (0\*4) + (0\*2) + (1\*1) = 16 + 8 + 1 = 25 🔁 What reverseBits function does: for (int i = 0; i < 32; i++) {  reversed = (reversed << 1) | ((num >> i) & 1);  }   * It shifts out each bit of num **starting from LSB**, and inserts it into the MSB side of reversed.   Think of it like flipping a coin 32 times and stacking them from left to right. 📊 Simulation (only the last 8 bits of reversed will matter): Let’s extract bits from **right to left** of num = 000...00011001:   | **i** | **(num >> i) & 1** | **reversed (binary)** | **reversed (decimal)** | | --- | --- | --- | --- | | 0 | 1 | 000...00000001 | 1 | | 1 | 0 | 000...00000010 | 2 | | 2 | 0 | 000...00000100 | 4 | | 3 | 1 | 000...00001001 | 9 | | 4 | 1 | 000...00010011 | 19 | | 5 | 0 | 000...00100110 | 38 | | 6 | 0 | 000...01001100 | 76 | | 7 | 0 | 000...10011000 | 152 | | 8-31 | 0s | right shift zeros only | final: 147 |   Wait — this gives **152**, not 147? 🤔  Let’s **actually go full 32-bits** and see the result:  After full 32-bit reversal, the **reversed binary** is:  10011000 00000000 00000000 00000000  But **we’re interpreting the final reversed number as an int**, which counts **all 32 bits**. ✅ Actual Output: cout << reversed << endl; // prints 147  So there’s a mismatch.  Let’s **double-check with code**:  int num = 25;  int reversed = reverseBits(num);  cout << reversed << endl;  The reversed **8 bits** of 25 (00011001) is 10011000 = **152**  BUT due to **bit reversal of all 32 bits**, reversed becomes a large number. So to get **147** as final output:  Let’s check the 32-bit reversal of:  00000000 00000000 00000000 00011001 // 25  Reversed:  10011000 00000000 00000000 00000000 // final  Now:  Binary: 10011000 00000000 00000000 00000000  Decimal: 1476395008  But that’s not 147 either. 💡 The Fix: You **must print the reversed result in 8-bit sense**, or mask it:  cout << (reversed >> 24) << endl;  This will give **actual 8-bit reversed form**, i.e.:  25 → 00011001  reversed → 10011000 → 152  But if your output is **147**, that means your original number is not 25, or the system is interpreting signed bits differently. |
| -1744830464 | |