|  |  |
| --- | --- |
| **Min Stack in C++** | |
| #include <iostream>  #include <stack>  #include <climits>  using namespace std;  class MinStack {  private:  stack<long long> st;  long long minVal;  public:  MinStack() {  minVal = INT\_MAX;  }  void push(int val) {  if (st.empty()) {  minVal = val;  st.push(0LL);  } else {  long long diff = val - minVal;  st.push(diff);  if (val < minVal) {  minVal = val;  }  }  }  void pop() {  long long rem = st.top();  st.pop();  if (rem < 0) {  minVal = minVal - rem;  }  }  int top() {  long long rem = st.top();  if (rem < 0) {  return static\_cast<int>(minVal);  } else {  return static\_cast<int>(minVal + rem);  }  }  int getMin() {  return static\_cast<int>(minVal);  }  };  int main() {  MinStack minStack;  minStack.push(2);  minStack.push(0);  minStack.push(3);  minStack.push(0);  cout << "Minimum value: " << minStack.getMin() << endl; // Should print 0  minStack.pop();  cout << "Minimum value: " << minStack.getMin() << endl; // Should print 0  minStack.pop();  cout << "Minimum value: " << minStack.getMin() << endl; // Should print 0  minStack.pop();  cout << "Minimum value: " << minStack.getMin() << endl; // Should print 2  return 0;  } | Core Logic Recap  * st stores **differences** between the current value and minVal. * If the pushed value is **less than** minVal, a **negative diff** is stored. This signals a **new min**. * When popping, if the top is negative, we **recalculate the previous min** using minVal - rem.  🧪 Test Input: minStack.push(2);  minStack.push(0);  minStack.push(3);  minStack.push(0);  pop() → getMin()  pop() → getMin()  pop() → getMin() 📋 Dry Run Table:  | **Operation** | **Stack (diffs)** | **minVal** | **Explanation** | | --- | --- | --- | --- | | push(2) | [0] | 2 | First element → diff is 0 | | push(0) | [0, -2] | 0 | 0 < 2 → store diff (-2), update minVal | | push(3) | [0, -2, 3] | 0 | 3 > 0 → store diff (3), minVal unchanged | | push(0) | [0, -2, 3, 0] | 0 | 0 = minVal → store diff (0), minVal unchanged | | pop() | [0, -2, 3] | 0 | popped 0, not negative → minVal stays | | getMin() | — | 0 |  | | pop() | [0, -2] | 0 | popped 3 (diff=3), not negative → minVal stays | | getMin() | — | 0 |  | | pop() | [0] | 2 | popped -2 → was a new min at the time → rollback | | getMin() | — | 2 |  |  ✅ Output: Minimum value: 0  Minimum value: 0  Minimum value: 0  Minimum value: 2 |
| Minimum value: 0  Minimum value: 0  Minimum value: 0  Minimum value: 2 | |