|  |  |
| --- | --- |
| **First non-repeating character in C++** | |
| #include <iostream>  #include <queue>  #include <unordered\_map>  using namespace std;  class FirstNonRepeatingCharacter {  public:  string FirstNonRepeating(string A) {  queue<char> q;  unordered\_map<char, int> hm;  string ans(A.length(), '#');  for (int i = 0; i < A.length(); i++) {  char c = A[i];  q.push(c);  hm[c]++;  while (!q.empty() && hm[q.front()] > 1) {  q.pop();  }  if (!q.empty()) {  ans[i] = q.front();  }  }  return ans;  }  };  int main() {  // Hardcoded input string  string A = "aabc";  // Create an instance of the FirstNonRepeatingCharacter class  FirstNonRepeatingCharacter solution;  // Call the FirstNonRepeating method and store the result  string result = solution.FirstNonRepeating(A);  // Print the result  cout << result << endl;  return 0;  } | **Code Summary:**   * Use a **queue** to maintain the order of characters. * Use a **hash map** (unordered\_map<char, int>) to count character occurrences. * At each step:   + Add current character to the queue.   + Increment its count.   + Remove characters from the front of the queue if their count > 1.   + The front of the queue (if any) is the current **first non-repeating** character.   **📦 Dry Run for A = "aabc"**   | **i** | **A[i]** | **Queue** | **Hash Map** | **First Non-Repeating** | **ans** | | --- | --- | --- | --- | --- | --- | | 0 | 'a' | a | a:1 | a | a | | 1 | 'a' | a a | a:2 | # (a is repeated) | a# | | 2 | 'b' | a a b → b | a:2, b:1 | b | a#b | | 3 | 'c' | b c | a:2, b:1, c:1 | b | a#bb |   **🧾 Final Output:**  a#bb  **✅ Explanation:**   * After 'a': only 'a' is in stream → 'a' * After second 'a': 'a' repeats → '#' * After 'b': 'b' is first non-repeating → 'b' * After 'c': 'b' is still non-repeating → 'b'   Top of Form  Bottom of Form |
| a#bb | |