|  |  |
| --- | --- |
| **Permutation of string in C++** | |
| #include <iostream>  #include <unordered\_map>  using namespace std;  void generate(int cs, int ts, unordered\_map<char, int>& fmap, string asf) {      if (cs > ts) {          cout << asf << endl;          return;      }      for (auto entry : fmap) {          char ch = entry.first;          int count = entry.second;            if (count > 0) {              fmap[ch]--;              generate(cs + 1, ts, fmap, asf + ch);              fmap[ch]++;          }      }  }  int main() {      string str = "abc";      unordered\_map<char, int> fmap;      for (char ch : str) {          fmap[ch]++;      }      generate(1, str.length(), fmap, "");      return 0;  } | **Goal:**  Generate **all permutations of "abc"** using recursion and a frequency map.  **🔧 Setup:**   * fmap: { a:1, b:1, c:1 } * ts = total size = 3 * cs = current size (starts from 1) * asf = answer so far   **🧾 Dry Run Table**   | **Call Stack** | **fmap (a,b,c)** | **asf** | **cs** | **Output?** | | --- | --- | --- | --- | --- | | generate(1, 3, {1,1,1}, "") |  |  |  |  | | └ a → generate(2, 3, {0,1,1}, "a") |  | "a" | 2 |  | | └ b → generate(3, 3, {0,0,1}, "ab") |  | "ab" | 3 |  | | └ c → generate(4, 3, {0,0,0}, "abc") |  | "abc" | 4 | ✅ Print | | └ c → backtrack to "ab" |  |  |  |  | | └ c → generate(3, 3, {0,1,0}, "ac") |  | "ac" | 3 |  | | └ b → generate(4, 3, {0,0,0}, "acb") |  | "acb" | 4 | ✅ Print | | └ b → backtrack to "a" |  |  |  |  | | └ b → generate(2, 3, {1,0,1}, "b") |  | "b" | 2 |  | | └ a → generate(3, 3, {0,0,1}, "ba") |  | "ba" | 3 |  | | └ c → generate(4, 3, {0,0,0}, "bac") |  | "bac" | 4 | ✅ Print | | └ c → generate(3, 3, {1,0,0}, "bc") |  | "bc" | 3 |  | | └ a → generate(4, 3, {0,0,0}, "bca") |  | "bca" | 4 | ✅ Print | | └ c → generate(2, 3, {1,1,0}, "c") |  | "c" | 2 |  | | └ a → generate(3, 3, {0,1,0}, "ca") |  | "ca" | 3 |  | | └ b → generate(4, 3, {0,0,0}, "cab") |  | "cab" | 4 | ✅ Print | | └ b → generate(3, 3, {1,0,0}, "cb") |  | "cb" | 3 |  | | └ a → generate(4, 3, {0,0,0}, "cba") |  | "cba" | 4 | ✅ Print | |
| Output:- cba  cab  bca  bac  acb  abc | |