|  |  |
| --- | --- |
| **Lexicographic order in C++** | |
| #include <iostream>  using namespace std;  void dfs(int i, int n) {  if (i > n) {  return;  }  cout << i << endl;  for (int j = 0; j < 10; j++) {  dfs(10 \* i + j, n);  }  }  int main() {  int n = 40;  for (int i = 1; i <= 9; i++) {  dfs(i, n);  }  return 0;  } | Lexicographical Order Output: This simulates how numbers are sorted like dictionary words:  1, 10, 100, ..., 11, 12, ..., 2, 20, 21, ..., 3, 30, ..., 4, 40, ..., 5, ..., 9  But only numbers ≤ 40 are printed. 📊 Dry Run Table (Partial for Clarity) Here’s a step-by-step snapshot of what’s happening:   | **Function Call** | **i** | **Output** | **Explanation** | | --- | --- | --- | --- | | dfs(1, 40) | 1 | ✅ 1 | Valid, print | | dfs(10, 40) | 10 | ✅ 10 | Valid, print | | dfs(100, 40) | >40 | ❌ | Stop recursion | | dfs(11, 40) | 11 | ✅ 11 | Continue same way | | dfs(12, 40) | 12 | ✅ 12 | ... | | ... | ... | ... |  | | dfs(19, 40) | 19 | ✅ 19 |  | | dfs(2, 40) | 2 | ✅ 2 | Start from next i | | dfs(20, 40) | 20 | ✅ 20 |  | | dfs(21, 40) | 21 | ✅ 21 |  | | ... | ... | ... |  | | dfs(40, 40) | 40 | ✅ 40 | Final valid number | | dfs(41, 40) | >40 | ❌ | Stop here | |
| Output:-  1  10  11  12  13  14  15  16  17  18  19  2  20  3  4  5  6  7  8  9 | |