|  |  |
| --- | --- |
| Print Boundary in C++ | |
| #include <iostream>  #include <vector>  using namespace std;  void printBoundary(vector<vector<int>>& mat) {  int n = mat.size();  int m = mat[0].size();  // Print top row  for (int j = 0; j < m; j++) {  cout << mat[0][j] << " ";  }  // Print right column (excluding the top and bottom elements already printed)  for (int i = 1; i < n; i++) {  cout << mat[i][m - 1] << " ";  }  // Print bottom row (excluding the bottom-right corner already printed)  if (n > 1) {  for (int j = m - 2; j >= 0; j--) {  cout << mat[n - 1][j] << " ";  }  }  // Print left column (excluding the top-left and bottom-left corners already printed)  if (m > 1) {  for (int i = n - 2; i > 0; i--) {  cout << mat[i][0] << " ";  }  }  }  int main() {  vector<vector<int>> mat = {  {1, 2, 3, 4, 5},  {6, 7, 8, 9, 10},  {11, 12, 13, 14, 15},  {16, 17, 18, 19, 20},  {21, 22, 23, 24, 25}  };  printBoundary(mat);  cout << endl;  return 0;  } | Input Matrix (5x5): [  [ 1, 2, 3, 4, 5 ],  [ 6, 7, 8, 9, 10 ],  [11, 12, 13, 14, 15 ],  [16, 17, 18, 19, 20 ],  [21, 22, 23, 24, 25 ]  ] ⏬ Step-by-step Dry Run Table:  | Step | Indices | Printed Values | | --- | --- | --- | | Top row | mat[0][0 to 4] | 1 2 3 4 5 | | Right column | mat[1 to 4][4] | 10 15 20 25 | | Bottom row | mat[4][3 to 0] | 24 23 22 21 | | Left column | mat[3 to 1][0] | 16 11 6 |  Dry Run Table  | Phase | Loop Variable(s) | Value Printed | | --- | --- | --- | | **Top Row** | j = 0 to 4 | 1 2 3 4 5 | | **Right Col** | i = 1 to 4 | 10 15 20 25 | | **Bottom Row** | j = 3 to 0 (reverse) | 24 23 22 21 | | **Left Col** | i = 3 to 1 (reverse) | 16 11 6 |  ✅ Final Output: 1 2 3 4 5 10 15 20 25 24 23 22 21 16 11 6 |
| 1 2 3 4 5 10 15 20 25 24 23 22 21 16 11 6 | |