# 2D Vector Operations in C++: Code Explanation

## Program Structure

This program demonstrates the usage of nested vectors (2D vectors) in C++ with various operations.

## Code Breakdown

### 1. Vector Initialization and Population

```
vector<vector<int>>v;    // Create a 2D vector
vector<int>v1;          // First row
v1.push_back(1);        // Add elements to first row
v1.push_back(2);
v1.push_back(3);
```

- Creates three separate vectors (v1, v2, v3) with different sizes
- v1 contains: [1,2,3]
- v2 contains: [4,5]
- v3 contains: [6,7,8,9]

### 2. Building the 2D Vector

```
v.push_back(v1);
v.push_back(v2);
v.push_back(v3);
```

Resulting 2D vector structure:

```
[
  [1,2,3],
```

```
  [4,5],
  [6,7,8,9]
]
```

## 3. Matrix Display

```
for(int i=0; i<v.size(); i++) {
    for(int j=0; j<v[i].size(); j++) {
        cout<<v[i][j]<<" ";
    }
    cout<<endl;
}
```

- Uses nested loops to print the 2D vector
- v.size() gives number of rows
- v[i].size() gives number of columns in each row

## 4. Element Modification

```
v[0][0] = 10;  // Changes 1 to 10 in first position
```

- Demonstrates direct element access and modification
- Uses array-like indexing syntax

## 5. Even-Odd Check

```
if(v[i][j] % 2 == 0) {
    cout << "Even number found: " << v[i][j] << "at  " << i << " , " << j << endl;
} else {
    cout << "Odd number found: " << v[i][j] << endl;
}
```

- Checks each element for even/odd property
- Prints element value along with its position for even numbers

## Key Points for Revision

- Vector of vectors allows rows of different lengths (jagged array)
- Use push_back() to add elements to vectors
- Access elements using double square brackets: v[i][j]
- Always check v[i].size() for inner loop as columns may vary
- Elements can be modified directly using index notation

## Time Complexity

- Vector creation: O(n) where n is total number of elements
- Displaying matrix: O(m×n) where m is rows and n is columns
- Element modification: O(1)
- Even-odd check: O(m×n)