### **Task 3: Advanced STL Operations**

**Goal:** Perform advanced operations such as transformations and conditional modifications on HMI states.

### **Scenario:**

The system needs to process HMI states based on certain conditions (e.g., hide all sliders during night mode).

### **Steps:**

1. **Manipulate Control States:**
   1. Use std::copy to create a backup of the control list.
   2. Use std::fill to set all states to "disabled" temporarily.
   3. Use std::generate to generate random states ("visible", "invisible", "disabled") for testing.
2. **Apply Transformations:**
   1. Use std::transform to change the state of all sliders to "invisible".
   2. Use std::replace to replace "disabled" with "enabled" for testing.
   3. Use std::remove\_if to filter out invisible controls from the list.
3. **Other Operations:**
   1. Use std::reverse to reverse the control order (e.g., for a debug layout).
   2. Use std::partition to group visible controls together.
4. **Output Results:**
   1. Show intermediate results after each operation.

**Program:**

#include <iostream>

#include <vector>

#include <string>

#include <algorithm>

#include <random>

using namespace std;

int main()

{

vector<string> controls = {"Slider1", "Slider2", "Button1", "Button2", "Slider3"};

vector<string> backupControls;

copy(controls.begin(), controls.end(), back\_inserter(backupControls));

cout << "Backup Control States: ";

for (const auto& control : backupControls)

{

cout << control << " ";

}

fill(controls.begin(), controls.end(), "disabled");

cout << "\nAfter filling all states with 'disabled': ";

for (const auto& control : controls)

{

cout << control << " ";

}

random\_device rd;

mt19937 gen(rd());

uniform\_int\_distribution<> dist(0, 2);

vector<string> states = {"visible", "invisible", "disabled"};

generate(controls.begin(), controls.end(), [&]() { return states[dist(gen)]; });

cout << "\nAfter generating random states: ";

for (const auto& control : controls)

{

cout << control << " ";

}

transform(controls.begin(), controls.end(), controls.begin(), [](string& state)

{

return state.find("Slider") != string::npos ? "invisible" : state;

});

cout << "\nAfter transforming all sliders to 'invisible': ";

for (const auto& control : controls)

{

cout << control << " ";

}

replace(controls.begin(), controls.end(), string("disabled"), string("enabled"));

cout << "\nAfter replacing 'disabled' with 'enabled': ";

for (const auto& control : controls)

{

cout << control << " ";

}

auto endIt = remove\_if(controls.begin(), controls.end(), [](const string& state)

{

return state == "invisible";

});

controls.erase(endIt, controls.end());

cout << "\nAfter removing invisible controls: ";

for (const auto& control : controls)

{

cout << control << " ";

}

reverse(controls.begin(), controls.end());

cout << "\nAfter reversing the control order: ";

for (const auto& control : controls)

{

cout << control << " ";

}

auto partitionIt = partition(controls.begin(), controls.end(), [](const string& state)

{

return state == "visible";

});

cout << "\nAfter partitioning visible controls: ";

for (const auto& control : controls)

{

cout << control << " ";

}

return 0;

}

Output:

Backup Control States: Slider1 Slider2 Button1 Button2 Slider3

After filling all states with 'disabled': disabled disabled disabled disabled disabled

After generating random states: disabled invisible invisible invisible visible

After transforming all sliders to 'invisible': disabled invisible invisible invisible visible

After replacing 'disabled' with 'enabled': enabled invisible invisible invisible visible

After removing invisible controls: enabled visible

After reversing the control order: visible enabled

After partitioning visible controls: visible enabled