Let’s consider a bubble sort program which sorts an array. Here’’s C++ implementation of bubble sort,

#include <bits/stdc++.h>

using namespace std;

void bubbleSort(int arr[], int n) {

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (arr[j] > arr[j + 1]) {

swap(arr[j], arr[j + 1]);

}

}

}

}

int main() {

int arr[] = {5, 4, 3, 2, 1};

int n = sizeof(arr) / sizeof(arr[0]);

bubbleSort(arr, n);

cout << "Sorted array: ";

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

cout << endl;

return 0;

}

Now here is three fuzzer to test this program,

**Mutation Fuzzer**

This fuzzer takes an existing input and mutates it randomly.

void mutate(int arr[], int n) {

srand(time(NULL));

int index1 = rand() % n;

int index2 = rand() % n;

int temp = arr[index1];

arr[index1] = arr[index2];

arr[index2] = temp;

}

void mutationFuzzer(){

int arr[] = {5, 2, 7, 3, 9};

int n = sizeof(arr)/sizeof(arr[0]);

cout << "Input array: ";

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

cout << endl;

for (int i = 0; i < 10; i++) {

mutate(arr, n);

cout << "Mutated array " << i+1 << ": ";

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

cout << endl;

}

}

This fuzzer randomly chooses two element from the array and swaps them to create a mutated array. It repeats 10 times for the same input array to generate 10 mutated arrays.

**Generation Fuzzer**

The generation fuzzer generates new inputs from scratch

void geneationFuzzer(){

srand(time(NULL));

for (int i = 0; i < 100; i++) {

int n = rand() % 20 + 1;

int arr[n];

for (int j = 0; j < n; j++) {

arr[j] = rand() % 100 + 1;

}

cout << "Generation Test " << i+1 << ": ";

for (int j = 0; j < n; j++) {

cout << arr[j] << " ";

}

cout << endl;

}

}

This fuzzer randomly generates an array of size between 1 and 20 with random values between 0 and 100. It then runs the bubble sort program on the generated array and prints the result repeating 100 times.

**Protocol-based Fuzzer**

This protocol-based fuzzer generates a different set of pairs each time the program is execute.

vector<pair<int, int>> generatePairs(int n) {

vector<pair<int, int>> pairs;

for (int i = 0; i < n - 1; i++) {

pairs.push\_back(make\_pair(i, i+1));

}

return pairs;

}

void shufflePairs(vector<pair<int, int>>& pairs) {

random\_device rd;

mt19937 g(rd());

shuffle(pairs.begin(), pairs.end(), g);

}

void protocolBasedFuzzer(){

int arr[] = {5, 2, 7, 3, 9};

int n = sizeof(arr)/sizeof(arr[0]);

cout << "Input Array: ";

for (int i = 0; i < n; i++) {

cout << arr[i] << " ";

}

cout << endl;

vector<pair<int, int>> pairs = generatePairs(n);

shufflePairs(pairs);

cout << "Protocol-Based Fuzzer: " << endl;

for (int i = 0; i < pairs.size(); i++) {

cout << arr[pairs[i].first] << " " << arr[pairs[i].second] << endl;

}

}