cpp

Copy code

#include <iostream>

#include <vector>

#include <algorithm>

#include <limits>

using namespace std;

const int MAX\_ARRAY\_SIZE = 1000;  // Max number of elements allowed

int main() {

    int n;

    // Ask the user for the number of elements in the array

    cout << "Enter the number of elements (max " << MAX\_ARRAY\_SIZE << "): ";

    while (true) {

        cin >> n;

        // Check if the user enters a valid integer and if it's within the allowed range

        if (cin.fail() || n <= 0 || n > MAX\_ARRAY\_SIZE) {

            cin.clear(); // Clear the error flag

            cin.ignore(numeric\_limits<streamsize>::max(), '\n'); // Ignore invalid input

            cout << "Invalid input. Please enter a positive integer up to " << MAX\_ARRAY\_SIZE << ": ";

        } else {

            break;

        }

    }

    vector<int> arr(n);

    // Input the array elements

    cout << "Enter the elements:" << endl;

    for (int i = 0; i < n; ++i) {

        while (true) {

            cin >> arr[i];

            // Check if the input is valid (an integer)

            if (cin.fail()) {

                cin.clear();

                cin.ignore(numeric\_limits<streamsize>::max(), '\n');

                cout << "Invalid input. Please enter an integer for element " << i + 1 << ": ";

            } else {

                break;

            }

        }

    }

    // Sort the array in ascending order

    sort(arr.begin(), arr.end());

    // Output the sorted array

    cout << "Sorted array: ";

    for (int i = 0; i < n; ++i) {

        cout << arr[i] << " ";

    }

    cout << endl;

    // Allow the user to query the array by index

    while (true) {

        int index;

        cout << "Enter an index to query (0 to " << n - 1 << "), or -1 to exit: ";

        cin >> index;

        // Exit the loop if the user enters -1

        if (index == -1) {

            break;

        }

        // Check if the index is valid

        if (index >= 0 && index < n) {

            cout << "Element at index " << index << ": " << arr[index] << endl;

        } else {

            cout << "Invalid index. Please try again." << endl;

        }

    }

    return 0;

}

**Key Security Improvements:**

1. **Maximum Array Size**:
   * The constant MAX\_ARRAY\_SIZE limits the maximum number of elements that can be entered. This helps prevent buffer overflow and excessive memory allocation.
2. **Input Validation for n**:
   * The program checks whether the user input for the number of elements (n) is valid and falls within the permissible range (positive integer ≤ MAX\_ARRAY\_SIZE).
3. **Input Validation for Array Elements**:
   * For each array element, the program ensures that only valid integers are entered. If a user enters an invalid input (non-integer), it asks the user to re-enter the value.
4. **Bounds Checking for Index Queries**:
   * Before accessing any element in the array, the program checks whether the user-provided index is within valid bounds (0 <= index < n).

**Example Run:**

vbnet

Copy code

Enter the number of elements (max 1000): 5

Enter the elements:

3 1 4 1 5

Sorted array: 1 1 3 4 5

Enter an index to query (0 to 4), or -1 to exit: 2

Element at index 2: 3

Enter an index to query (0 to 4), or -1 to exit: 10

Invalid index. Please try again.

Enter an index to query (0 to 4), or -1 to exit: -1

**Benefits:**

* **Buffer Overflow Prevention**: By using vector<int> and restricting n to a reasonable range (MAX\_ARRAY\_SIZE), we avoid risks of buffer overflows.

**Robust Input Handling**: The program validates input for both the number of elements and the array values, ensuring the program doesn't crash or behave unpredictably when faced with invalid input.