**Here's a C++ program that sorts an array of integers and allows indexed queries:**

C++

#include <iostream>

#include <algorithm>

using namespace std;

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

    sort(arr, arr + n);

}

int queryByIndex(int arr[], int n, int index) {

    if (index >= 0 && index < n) {

        return arr[index];

    } else {

        cout << "Invalid index. Please enter a valid index between 0 and " << n - 1 << endl;

        return -1;

    }

}

int main() {

    int n;

    cout << "Enter the number of elements: ";

    cin >> n;

    int arr[n];

    cout << "Enter the elements: ";

    for (int i = 0; i < n; i++) {

        cin >> arr[i];

    }

    sortArray(arr, n);

    cout << "Sorted array: ";

    for (int i = 0; i < n; i++) {

        cout << arr[i]

<< " ";

    }

    cout << endl;

    int queryIndex;

    while (true) {

        cout << "Enter the index to query (or -1 to exit): ";

        cin >> queryIndex;

        if (queryIndex == -1) {

            break;

        }

        int result = queryByIndex(arr, n, queryIndex);

        if (result != -1) {

            cout << "Element at index " << queryIndex << ": " << result << endl;

        }

    }

    return 0;

}

Use code with caution.

**Explanation:**

1. **sortArray function:**
   * Takes an integer array arr and its size n as input.
   * Uses the std::sort function from the <algorithm> library to sort the array in ascending order.
2. **queryByIndex function:**
   * Takes the sorted array arr, its size n, and the query index index as input.
   * Checks if the index is valid.
   * If valid, returns the element at the specified index.
   * If invalid, prints an error message and returns -1.
3. **main function:**
   * Prompts the user to enter the number of elements n.
   * Reads n integers into the array arr.
   * Calls the sortArray function to sort the array.
   * Prints the sorted array.
   * Enters a loop to repeatedly prompt the user for a query index.
   * Exits the loop when the user enters -1.
   * Calls the queryByIndex function to get the element at the specified index.
   * Prints the result or an error message.

**Key Points:**

* **Efficient Sorting:** The std::sort function is a highly optimized sorting algorithm, usually implemented as Introsort.
* **Clear Input/Output:** The code provides clear prompts and informative messages to guide the user.
* **Error Handling:** The queryByIndex function handles invalid index inputs gracefully.
* **User-Friendly Interface:** The loop allows for multiple queries without restarting the program.

This program provides a robust and efficient solution to the given problem.