Iterative Binary search in C++

```cpp
#include <iostream>
#include <vector>

using namespace std;

int binsearch(const vector<int>& arr, int x) {
    int low = 0, high = arr.size() - 1;

    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == x) {
            return mid;
        } else if (arr[mid] > x) {
            high = mid - 1;
        } else {
            low = mid + 1;
        }
    }
    return -1;
}

int main() {
    vector<int> arr = {3, 5, 7, 8, 9};
    cout << binsearch(arr, 8) << endl;
    return 0;
}
```

Input Details

- arr = {3, 5, 7, 8, 9}
- x = 8

Binary Search Table

Step	low	high	mid	arr[mid]	Comparison	Action
1	0	4	(0+4)/2 = 2	7	7 < 8 → false	low = mid + 1 → 3
2	3	4	(3+4)/2 = 3	8	8 == 8 → true	**Return 3**

Output

3

3

Binary Recursive in C++

```cpp
#include <iostream>
#include <vector>

using namespace std;

int binsearch(const vector<int>& arr, int low, int high, int x) {
    if (low > high) {
        return -1;
    }
    int mid = (low + high) / 2;
    if (arr[mid] == x) {
        return mid;
    } else if (arr[mid] > x) {
        return binsearch(arr, low, mid - 1, x);
    } else {
        return binsearch(arr, mid + 1, high, x);
    }
}

int main() {
    vector<int> arr = {3, 5, 7, 8, 9, 11, 45, 76};
    int result = binsearch(arr, 0, arr.size() - 1, 11);
    cout << result << endl;
    return 0;
}
```

Here's a **tabular dry run** of the **recursive binary search** code for:

arr = {3, 5, 7, 8, 9, 11, 45, 76}
x = 11

🧮 Dry Run Table

Call #	low	high	mid = (low+high)/2	arr[mid]	Comparison	Action
1	0	7	(0+7)/2 = 3	8	8 < 11	Search right → low = mid+1 = 4
2	4	7	(4+7)/2 = 5	11	11 == 11	**Found** → return 5

✅ Output

5

5

Pair with Given Sum in C++

```cpp
#include <iostream>
#include <vector>

using namespace std;

bool pairWithGivenSum(const vector<int>& arr, int x)
{
    int left = 0, right = arr.size() - 1;

    while (left < right) {
        if (arr[left] + arr[right] == x) {
            return true;
        } else if (arr[left] + arr[right] > x) {
            right--;
        } else {
            left++;
        }
    }
    return false;
}

int main() {
    vector<int> arr = {10, 7, 8, 20, 12};
    int x = 32;

    cout << std::boolalpha << pairWithGivenSum(arr,
x) << endl;
    return 0;
}
```

Dry Run After Sorting

Sorted array: {7, 8, 10, 12, 20}
Target x = 32

left (val)	right (val)	Sum	Action
7	20	27	Increase left
8	20	28	Increase left
10	20	30	Increase left
12	20	32	🎯 Match → return true

✅ **Output:**

true

true

Peak element in C++

```
#include <iostream>
#include <vector>

using namespace std;

int findPeakElement(const vector<int>& arr) {
    int low = 0, high = arr.size() - 1;

    while (low <= high) {
        int mid = (low + high) / 2;

        if ((mid == 0 || arr[mid - 1] <= arr[mid])
                && (mid == arr.size() - 1 || arr[mid + 1] <= arr[mid])) {
            return mid;
        }

        if (mid > 0 && arr[mid - 1] >= arr[mid]) {
            high = mid - 1;
        } else {
            low = mid + 1;
        }
    }

    return -1; // Peak element not found
}

int main() {
    vector<int> arr = {10, 7, 8, 20, 12};
    cout << findPeakElement(arr) << endl;
    return 0;
}
```

Dry Run Table:

Iteration	low	high	mid	arr[mid-1]	arr[mid]	arr[mid+1]	Condition Met	Action
1	0	4	2	7	8	20	Right neighbor > mid	low = mid + 1 = 3
2	3	4	3	8	20	12	Both neighbors ≤ mid → **peak found!**	Return 3

✅ **Output:**

3

3

Sqrt in C++

```cpp
#include <iostream>

using namespace std;

int sqrt(int x) {
    if (x == 0 || x == 1) {
        return x;
    }

    int low = 1, high = x, ans = 0;
    while (low <= high) {
        int mid = low + (high - low) / 2;
        long long mSqr = (long long) mid * mid; // Use long long to avoid integer overflow

        if (mSqr == x) {
            return mid;
        } else if (mSqr > x) {
            high = mid - 1;
        } else {
            low = mid + 1;
            ans = mid;
        }
    }
    return ans;
}

int main() {
    cout << sqrt(37) << endl;
    return 0;
}
```

Dry Run Table:

Iteration	low	high	mid	mid*mid	ans	Action
1	1	37	19	361	0	361 > 37 → high = mid - 1 = 18
2	1	18	9	81	0	81 > 37 → high = mid - 1 = 8
3	1	8	4	16	0	16 < 37 → ans = 4, low = mid + 1 = 5
4	5	8	6	36	4	36 < 37 → ans = 6, low = mid + 1 = 7
5	7	8	7	49	6	49 > 37 → high = mid - 1 = 6
End	7	6	-	-	6	Loop ends since low > high

✅ Final Result:

6

6