### 1. Binary Search with a Sorted Array of Pairs (by First Element)

cpp

#include <iostream>

#include <vector>

using namespace std;

int binarySearchPairFirst(vector<pair<int, int>>& arr, int target) {

int left = 0, right = arr.size() - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid].first == target) return mid;

if (arr[mid].first < target) left = mid + 1;

else right = mid - 1;

}

return -1;

}

int main() {

vector<pair<int, int>> arr = {{1, 10}, {2, 20}, {3, 30}, {4, 40}, {5, 50}};

int target = 3;

int result = binarySearchPairFirst(arr, target);

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

return 0;

}

### 2. Binary Search with a Sorted Array of Tuples

cpp

#include <iostream>

#include <tuple>

#include <vector>

using namespace std;

int binarySearchTuple(vector<tuple<int, string>>& arr, int target) {

int left = 0, right = arr.size() - 1;

cpp

while (left <= right) {

int mid = left + (right - left) / 2;

if (get<0>(arr[mid]) == target) return mid;

if (get<0>(arr[mid]) < target) left = mid + 1;

else right = mid - 1;

}

return -1;

}

int main() {

vector<tuple<int, string>> arr = {make\_tuple(1, "One"), make\_tuple(2, "Two"), make\_tuple(3, "Three"), make\_tuple(4, "Four"), make\_tuple(5, "Five")};

int target = 4;

int result = binarySearchTuple(arr, target);

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

return 0;

}

### 3. Binary Search with a Sorted Array of Structs (by ID)

cpp

#include <iostream>

#include <string>

using namespace std;

struct Book {

int id;

string title;

};

int binarySearchBookById(Book arr[], int size, int targetId) {

int left = 0, right = size - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid].id == targetId) return mid;

if (arr[mid].id < targetId) left = mid + 1;

else right = mid - 1;

}

return -1;

}

int main() {

Book arr[] = { {1, "1984"}, {2, "Brave New World"}, {3, "Fahrenheit 451"}, {4, "The Great Gatsby"}, {5, "Moby Dick"} };

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

int targetId = 3;

int result = binarySearchBookById(arr, size, targetId);

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

return 0;

}

### 4. Binary Search with a Sorted Array of Pointers

cpp

#include <iostream>

using namespace std;

int binarySearchPointer(int\* arr[], int size, int target) {

int left = 0, right = size - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (\*arr[mid] == target) return mid;

if (\*arr[mid] < target) left = mid + 1;

else right = mid - 1;

}

return -1;

}

int main() {

int a = 1, b = 2, c = 3, d = 4, e = 5;

int\* arr[] = {&a, &b, &c, &d, &e};

int target = 3;

int result = binarySearchPointer(arr, 5, target);

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

return 0;

}

### 5. Binary Search with a Sorted Array of Unique Elements

cpp

#include <iostream>

using namespace std;

int binarySearchUnique(int arr[], int size, int target) {

int left = 0, right = size - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid] == target) return mid;

if (arr[mid] < target) left = mid + 1;

else right = mid - 1;

}

return -1;

}

int main() {

int arr[] = {1, 3, 5, 7, 9};

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

int target = 5;

int result = binarySearchUnique(arr, size, target);

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

return 0;

}

### 6. Binary Search with a Sorted Array of Negative and Positive Integers

cpp

#include <iostream>

using namespace std;

int binarySearchMixed(int arr[], int size, int target) {

int left = 0, right = size - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid] == target) return mid;

if (arr[mid] < target) left = mid + 1;

else right = mid - 1;

}

return -1;

}

int main() {

int arr[] = {-10, -5, 0, 5, 10};

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

int target = 0;

int result = binarySearchMixed(arr, size, target);

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

return 0;

}

### 7. Binary Search with a Sorted Array of Large Integers

cpp

#include <iostream>

using namespace std;

int binarySearchLarge(int arr[], int size, long long target) {

int left = 0, right = size - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid] == target) return mid;

if (arr[mid] < target) left = mid + 1;

else right = mid - 1;

}

return -1;

}

int main() {

long long arr[] = {1000000000, 2000000000, 3000000000, 4000000000, 5000000000};

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

long long target = 3000000000;

int result = binarySearchLarge(arr, size, target);

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

return 0;

}

### 8. Binary Search with a Sorted Array of Mixed Types (using void pointers)

cpp

#include <iostream>

#include <cstring>

using namespace std;

int binarySearchMixedTypes(void\* arr[], int size, void\* target, int (cmp)(void, void\*)) {

int left = 0, right = size - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (cmp(arr[mid], target) == 0) return mid;

if (cmp(arr[mid], target) < 0) left = mid + 1;

else right = mid - 1;

}

return -1;

}

int compareInt(void\* a, void\* b) {

return (int)a - (int)b;

}

int main() {

int a = 1, b = 2, c = 3, d = 4, e = 5;

void\* arr[] = {&a, &b, &c, &d, &e};

int target = 3;

int result = binarySearchMixedTypes(arr, 5, &target, compareInt);

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

return 0;

}

### 9. Binary Search with a Sorted Array of Structs (by Age)

cpp

#include <iostream>

#include <string>

using namespace std;

struct Student {

int age;

string name;

};

int binarySearchStudentByAge(Student arr[], int size, int targetAge) {

int left = 0, right = size - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid].age == targetAge) return mid;

if (arr[mid].age < targetAge) left = mid + 1;

else right = mid - 1;

}

return -1;

}

int main() {

Student arr[] = { {20, "Alice"}, {22, "Bob"}, {24, "Charlie"}, {26, "David"}, {28, "Eve"} };

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

int targetAge = 24;

int result = binarySearchStudentByAge(arr, size, targetAge);

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

return 0;

}

### 10. Binary Search with a Sorted Array of Pointers to Structs

cpp

#include <iostream>

#include <string>

using namespace std;

struct Employee {

int id;

string name;

};

int binarySearchEmployee(Employee\* arr[], int size, int targetId) {

int left = 0, right = size - 1;

while (left <= right) {

int mid = left + (right - left) / 2;

if (arr[mid]->id == targetId) return mid;

if (arr[mid]->id < targetId) left = mid + 1;

else right = mid - 1;

}

return -1;

}

int main() {

Employee emp1 = {1, "John"}, emp2 = {2, "Jane"}, emp3 = {3, "Doe"};

Employee\* arr[] = {&emp1, &emp2, &emp3};

int targetId = 2;

int result = binarySearchEmployee(arr, 3, targetId);

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

return 0;

```cpp

}