EJERCICIO 1 — Buscar Libro por ISBN (Búsqueda Binaria)

Planteamiento

Se tiene una lista ORDENADA por ISBN.
Se desea buscar si existe el ISBN **978014312**.

Razonamiento

Como el arreglo está ordenado, aplicamos **búsqueda binaria**, dividiendo el arreglo a la mitad en cada paso.

Código C++

```
#include <iostream>
#include <cstring>
using namespace std;

struct Libro {
    int isbn;
    char titulo[50];
    char autor[50];
    int anio;
};

int buscarISBN(Libro* arr, int n, int key) {
    int low = 0, high = n - 1;
    while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid].isbn == key) return mid;
        if (arr[mid].isbn < key) low = mid + 1;
        else high = mid - 1;
    }
    return -1;
}

int main() {
    int n;
    cin >> n;

    Libro* L = new Libro[n];

    for (int i = 0; i < n; i++) {
        cin >> L[i].isbn >> L[i].anio;
        cin.ignore();
        cin.getline(L[i].titulo, 50);
        cin.getline(L[i].autor, 50);
    }

    int key = 978014312;
    int pos = buscarISBN(L, n, key);
```

```
    if (pos != -1) {
        cout << "Posicion: " << pos << "\n";
        cout << L[pos].titulo << "\n" << L[pos].autor <<
endl;
    } else {
        cout << "Libro no encontrado\n";
    }

    delete[] L;
    return 0;
}
```

Ejemplo de entrada

```
3
978013110 1978
The C Programming Language
Kernighan & Ritchie
978014312 2010
Norwegian Wood
Haruki Murakami
978067978 1993
The Firm
John Grisham
```

Ejemplo de salida

```
Posicion: 1
Norwegian Wood
Haruki Murakami
```

EJERCICIO 2 — Buscar Estudiante por Matrícula (Búsqueda Binaria)

Planteamiento

Lista ORDENADA por matrícula (número entero).
Se debe buscar al estudiante con matrícula **202310045**.

Razonamiento

Aplicamos búsqueda binaria sobre el campo matrícula.

Código C++

```
#include <iostream>
#include <cstring>
using namespace std;

struct Alumno {
    int matricula;
    char nombre[40];
    char carrera[40];
    float promedio;
};
```

```
int buscarMat(Alumno* arr, int n, int key){
    int low = 0, high = n-1;
    while(low <= high){
        int mid = (low + high)/2;
        if(arr[mid].matricula == key) return mid;
        if(arr[mid].matricula < key) low = mid + 1;
        else high = mid - 1;
    }
    return -1;
}

int main(){
    int n;
    cin >> n;

    Alumno* A = new Alumno[n];

    for(int i=0;i<n;i++){
        cin >> A[i].matricula >> A[i].promedio;
        cin.ignore();
        cin.getline(A[i].nombre,40);
        cin.getline(A[i].carrera,40);
    }

    int key = 202310045;
    int pos = buscarMat(A,n,key);

    if(pos!=-1){
        cout<<"Encontrado\n";
        cout<<A[pos].nombre<<"\n"<<A[pos].carrera<<"\n";
        cout<<A[pos].promedio;
    } else {
        cout<<"Alumno no registrado";
    }

    delete[] A;
    return 0;
}
```

Ejemplo de entrada

```
3
202110010 8.5
Ana Torres
Administracion
202310045 9.2
Luis Paredes
Software
202410150 7.8
Maria Ruiz
```

```
Contabilidad
```

Salida

```
Encontrado
Luis Paredes
Software
9.2
```

EJERCICIO 3 — Buscar Película por Año (Búsqueda Binaria)

Planteamiento

Lista ordenada por año.
Buscar la película del año **1994**.

Razonamiento

Como los años están ordenados, se usa búsqueda binaria.

Código C++

```
#include <iostream>
#include <cstring>
using namespace std;

struct Pelicula {
    int anio;
    char titulo[50];
    float rating;
};

int buscarAnio(Pelicula* arr, int n, int key){
    int low=0, high=n-1;
    while(low<=high){
        int mid=(low+high)/2;
        if(arr[mid].anio==key) return mid;
        if(arr[mid].anio < key) low=mid+1;
        else high=mid-1;
    }
    return -1;
}

int main(){
    int n;
    cin>>n;

    Pelicula* P=new Pelicula[n];

    for(int i=0;i<n;i++){
        cin>>P[i].anio>>P[i].rating;
```

```
        cin.ignore();
        cin.getline(P[i].titulo,50);
    }

    int key=1994;
    int pos=buscarAnio(P,n,key);

    if(pos!=-1){
        cout<<"Pelicula: "<<P[pos].titulo<<"\n";
        cout<<"Rating: "<<P[pos].rating<<"\n";
    } else {
        cout<<"No existe pelicula de ese anio";
    }

    delete[] P;
    return 0;
}
```

Ejemplo de entrada

```
3
1972 9.2
The Godfather
1994 9.3
Forrest Gump
2008 9.0
The Dark Knight
```

Salida

```
Pelicula: Forrest Gump
Rating: 9.3
```

EJERCICIO 4 — Buscar Empleado por Sueldo (Búsqueda Binaria)

Planteamiento

Una empresa almacena sueldos ORDENADOS de menor a mayor.
Cada registro tiene:

- nombre
- cargo
- sueldo

Se desea buscar al empleado cuyo sueldo sea **1250.50**.

Razonamiento

Como el arreglo está ordenado por sueldo, se usa **búsqueda binaria** para optimizar el tiempo.

Código C++

```
#include <iostream>
#include <cstring>
using namespace std;

struct Empleado {
    float sueldo;
    char nombre[40];
    char cargo[40];
};

int buscarSueldo(Empleado* arr, int n, float key){
    int low=0, high=n-1;
    while(low <= high){
        int mid = (low+high)/2;
        if(arr[mid].sueldo == key) return mid;
        if(arr[mid].sueldo < key) low = mid + 1;
        else high = mid - 1;
    }
    return -1;
}

int main(){
    int n;
    cin >> n;

    Empleado* E = new Empleado[n];

    for(int i=0; i<n; i++){
        cin >> E[i].sueldo;
        cin.ignore();
        cin.getline(E[i].nombre,40);
        cin.getline(E[i].cargo,40);
    }

    float key = 1250.50;
    int pos = buscarSueldo(E,n,key);

    if(pos != -1){
        cout << "Encontrado:\n";
        cout << E[pos].nombre << "\n" << E[pos].cargo <<
endl;
    } else {
        cout << "No existe un empleado con ese sueldo.";
    }
```

```
    delete[] E;
    return 0;
}
```

Ejemplo de Entrada

```
3
900.00
Ana Perez
Secretaria
1250.50
Mario Torres
Analista
1500.00
Jorge Ruiz
Supervisor
```

Salida

```
Encontrado:
Mario Torres
Analista
```

EJERCICIO 5 — Buscar Curso por Número de Créditos (Búsqueda Binaria)

Planteamiento

Lista de cursos ORDENADA por número de créditos:

Cada curso contiene:

- nombre
- código
- créditos

Buscar el curso con **3 créditos**.

Razonamiento

Los créditos están ordenados, así que la búsqueda binaria es la más eficiente.

Código C++

```
#include <iostream>
#include <cstring>
using namespace std;

struct Curso {
    int creditos;
    char codigo[20];
    char nombre[50];
};

int buscarCred(Curso* arr, int n, int key){
    int low=0, high=n-1;
    while(low<=high){
        int mid=(low+high)/2;
        if(arr[mid].creditos == key) return mid;
        if(arr[mid].creditos < key) low=mid+1;
        else high=mid-1;
    }
    return -1;
}

int main(){
    int n;
    cin>>n;

    Curso* C=new Curso[n];

    for(int i=0;i<n;i++){
        cin>>C[i].creditos;
        cin.ignore();
        cin.getline(C[i].codigo,20);
        cin.getline(C[i].nombre,50);
    }

    int key=3;

    int pos=buscarCred(C,n,key);

    if(pos!=-1){
        cout<<"Curso encontrado:\n";
        cout<<C[pos].codigo<<"\n"<<C[pos].nombre<<"\n";
    } else {
        cout<<"No hay curso de 3 creditos";
    }

    delete[] C;
    return 0;
}
```

Ejemplo de Entrada

```
3
2
MAT101
Calculo I
3
INF203
Estructura de Datos
4
ELE450
Circuitos II
```

Salida

```
Curso encontrado:
INF203
Estructura de Datos
```

EJERCICIO 6 — Buscar Ciudad por Población (Búsqueda Binaria)

Planteamiento

Una lista ordenada de ciudades por número de habitantes.
Buscar la ciudad con **850000 habitantes**.

Razonamiento

La población está ordenada → aplicar búsqueda binaria.

Código C++

```
#include <iostream>
#include <cstring>
using namespace std;

struct Ciudad {
    long poblacion;
    char nombre[40];
    char pais[40];
};

int buscarPob(Ciudad* arr, int n, long key){
    int low=0, high=n-1;
    while(low<=high){
        int mid=(low+high)/2;
        if(arr[mid].poblacion==key) return mid;
        if(arr[mid].poblacion < key) low=mid+1;
        else high=mid-1;
    }
```

```
    return -1;
}

int main(){
    int n;
    cin>>n;

    Ciudad* C=new Ciudad[n];

    for(int i=0;i<n;i++){
        cin>>C[i].poblacion;
        cin.ignore();
        cin.getline(C[i].nombre,40);
        cin.getline(C[i].pais,40);
    }

    long key=850000;

    int pos=buscarPob(C,n,key);

    if(pos!=-1){
        cout<<"Ciudad encontrada:\n";
        cout<<C[pos].nombre<<"\n"<<C[pos].pais<<"\n";
    } else {
        cout<<"No existe ciudad con esa poblacion";
    }

    delete[] C;
    return 0;
}
```

Ejemplo de Entrada

```
3
500000
Cuenca
Ecuador
850000
Manta
Ecuador
1200000
Guayaquil
Ecuador
```

Salida

```
Ciudad encontrada:
Manta
Ecuador
```

EJERCICIO 7 — Buscar Producto por Stock (Búsqueda Binaria)

Planteamiento

Una tienda tiene una lista de productos ordenados por **cantidad en stock**.
Cada producto contiene:

- nombre
- categoría
- stock (ordenado ascendente)

Se desea buscar el producto con **50 unidades** en stock.

Razonamiento

El stock está ordenado, así que se usa **búsqueda binaria** para encontrarlo rápidamente.

Código C++

```
#include <iostream>
#include <cstring>
using namespace std;

struct Producto {
    int stock;
    char nombre[40];
    char categoria[40];
};

int buscarStock(Producto* P, int n, int key){
    int low=0, high=n-1;
    while(low<=high){
        int mid=(low+high)/2;

        if(P[mid].stock == key) return mid;
        if(P[mid].stock < key) low = mid + 1;
        else high = mid - 1;
    }
    return -1;
}

int main(){
    int n;
    cin >> n;

    Producto* P = new Producto[n];
```

```
    for(int i=0; i<n; i++){
        cin >> P[i].stock;
        cin.ignore();
        cin.getline(P[i].nombre,40);
        cin.getline(P[i].categoria,40);
    }

    int key = 50;
    int pos = buscarStock(P, n, key);

    if(pos != -1){
        cout << "Producto encontrado:\n";
        cout << P[pos].nombre << endl;
        cout << P[pos].categoria << endl;
    } else {
        cout << "No existe producto con stock de 50.";
    }

    delete[] P;
    return 0;
}
```

Ejemplo Entrada

```
3
20
Leche Entera
Lacteos
50
Arroz Premium
Granos
200
Aceite Vegetal
Abarrotes
```

Salida

```
Producto encontrado:
Arroz Premium
Granos
```

EJERCICIO 8 — Buscar Libro por Número de Páginas (Búsqueda Binaria)

Planteamiento

Una biblioteca guarda libros ordenados por **número de páginas**.
Cada registro contiene:

- título
- autor
- páginas

Se quiere buscar el libro de **420 páginas**.

Razonamiento

Dado que las páginas están ordenadas, la búsqueda binaria es la ideal.

Código C++

```
#include <iostream>
#include <cstring>
using namespace std;

struct Libro {
    int paginas;
    char titulo[50];
    char autor[50];
};

int buscarPag(Libro* L, int n, int key){
    int low=0, high=n-1;
    while(low<=high){
        int mid=(low+high)/2;

        if(L[mid].paginas == key) return mid;
        if(L[mid].paginas < key) low = mid + 1;
        else high = mid - 1;
    }
    return -1;
}

int main(){
    int n;
    cin >> n;

    Libro* L = new Libro[n];

    for(int i=0; i<n; i++){
        cin >> L[i].paginas;
        cin.ignore();
        cin.getline(L[i].titulo,50);
        cin.getline(L[i].autor,50);
    }

    int key = 420;
    int pos = buscarPag(L, n, key);
```

```
    if(pos != -1){
        cout << "Libro encontrado:\n";
        cout << L[pos].titulo << endl;
        cout << L[pos].autor << endl;
    } else {
        cout << "No existe libro con 420 páginas.";
    }

    delete[] L;
    return 0;
}
```

Ejemplo Entrada

```
3
120
El Principito
Antoine de Saint-Exupery
420
El Hobbit
J.R.R. Tolkien
800
It
Stephen King
```

Salida

```
Libro encontrado:
El Hobbit
J.R.R. Tolkien
```

EJERCICIO 9 — Buscar Estudiante por Nota Final (Búsqueda Binaria)

Planteamiento

Una lista de estudiantes ordenada por nota final (0–100).
Cada uno contiene:

- nombre
- carrera
- nota

Buscar la nota **88**.

Razonamiento

Notas ordenadas → usar búsqueda binaria.

Código C++

```
#include <iostream>
#include <cstring>
using namespace std;

struct Alumno {
    int nota;
    char nombre[40];
    char carrera[40];
};

int buscarNota(Alumno* A, int n, int key){
    int low=0, high=n-1;

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

        if(A[mid].nota == key) return mid;
        if(A[mid].nota < key) low=mid+1;
        else high=mid-1;
    }
    return -1;
}

int main(){
    int n;
    cin >> n;

    Alumno* A = new Alumno[n];

    for(int i=0; i<n; i++){
        cin >> A[i].nota;
        cin.ignore();
        cin.getline(A[i].nombre,40);
        cin.getline(A[i].carrera,40);
    }

    int key = 88;
    int pos = buscarNota(A,n,key);

    if(pos != -1){
        cout << "Alumno encontrado:\n";
        cout << A[pos].nombre << endl;
        cout << A[pos].carrera << endl;
    } else {
        cout << "No existe estudiante con nota 88";
```

```
    }

    delete[] A;
    return 0;
}
```

Ejemplo Entrada

```
3
60
Luis Vera
Administracion
88
Maria Loja
Sistemas
99
Carlos Suarez
Medicina
```

Salida

```
Alumno encontrado:
Maria Loja
Sistemas
```

EJERCICIO 10 — Buscar Vehículo por Kilometraje (Búsqueda Binaria)

Planteamiento

Una concesionaria almacena vehículos ordenados por kilometraje:

- marca
- modelo
- km (ordenado de menor a mayor)

Se desea buscar **45000 km**.

Razonamiento

Como los kilómetros están ordenados:
→ búsqueda binaria.

Código C++

```
#include <iostream>
#include <cstring>
using namespace std;

struct Vehiculo {
    long km;
    char marca[40];
    char modelo[40];
};

int buscarKM(Vehiculo* V, int n, long key){
    int low=0, high=n-1;

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

        if(V[mid].km == key) return mid;
        if(V[mid].km < key) low = mid + 1;
        else high = mid - 1;
    }

    return -1;
}

int main(){
    int n;
    cin>>n;

    Vehiculo* V = new Vehiculo[n];

    for(int i=0; i<n; i++){
        cin>>V[i].km;
        cin.ignore();
        cin.getline(V[i].marca,40);
        cin.getline(V[i].modelo,40);
    }

    long key = 45000;

    int pos = buscarKM(V,n,key);

    if(pos != -1){
        cout<<"Vehiculo encontrado:\n";
        cout<<V[pos].marca<<endl;
        cout<<V[pos].modelo<<endl;
    } else {
        cout<<"No existe vehiculo con 45000 km";
    }

    delete[] V;
    return 0;
```

```
}
```

Ejemplo Entrada

```
3
10000
Kia
Rio
45000
Toyota
Corolla
78000
Hyundai
Accent
```

Salida

```
Vehiculo encontrado:
Toyota
Corolla
```