## Búsqueda Secuencial

Consiste en revisar cada elemento hasta encontrar el dato buscado. Cuando se habla de búsqueda en vectores debe de distinguirse entre vectores desordenados y vectores ordenados.

### En arreglos desordenados

La búsqueda secuencial en arreglos desordenados consiste básicamente, en recorrer el arreglo de izquierda a derecha hasta que encuentre el elemento buscado. Normalmente cuando una función de búsqueda concluya con éxito, interesa conocer si el elemento pertenece o no al vector.

Sea A el vector

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 8 | 40 | 26 | 12 | 1 | 4 | 14 |

Elemento a buscar 26

Comparamos cada elemento del vector con el elemento a buscar (26), como encuentra el elemento a buscar devuelve verdad, ya que 26 pertenece al vector.

**Algoritmo**

Para cada elemento del vector, determina si es igual al elemento buscado, esta búsqueda es realizado por la función buscaVectorDesordenado, que devuelve verdad si elemento pertenece al vector y falso si elemento no pertenece al vector.

*Especificación del TAD y los algoritmos*

**Especificación** BUSCA

**variable**

nro: entero.

**métodos**

menu : no retorna valor.

ingresarDatos() : no retorna valor.

mostrarDatos() : no retorna valor.

registrarDatos(a, n, x) : no retorna valor.

*buscaVectorDesordenado*(a,n, ele) : retorna valor.

visualizarVector(a, n) : no retorna valor.

**Significado**

*menu* muestra las opciones a escoger.

*ingresarDatos* ingresa datos al vector.

*mostrarDatos* muestra los datos del vector.

*registrarDatos* registra los datos de *x* en el vector *a*.

*buscaVectorDesordenado* tiene como precondición al vector a, su número de elementos y elemento a buscar, y como postcondición devuelve verdad si elemento pertenece al vector y falso en caso contrario.

mostrarVector muestra el vector ordenado.

**Fin\_especificacion**

**Algoritmo**

Funcion buscaVectorDesordenado (a , n, ele): logico

Entero: i

Logico r

i = 0

Mientras (a[i].num != ele && i < n) Hacer

i = i + 1

Fin\_mientras

r = (a[i].num == ele)

retornar r

Fin\_funcion

**Resultados de la corrida de prueba del algoritmo**

**n=7**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 8 | 40 | 26 | 12 | 1 | 4 | 14 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

ele=26

|  |  |  |  |
| --- | --- | --- | --- |
| índice | valores | Resultado de la expresión lógica | Resultados del cuerpo del bucle |
| a[i]!=26 y i<7 | i=i+1 |
| 0 | a[0]=8; 8!=26 (v) y 0<7(v) | true | 1 |
| 1 | a[1]=40; 40!=26 (v) y 1<7(v) | true | 2 |
| 2 | a[2]=26; 26!=26 (f) y 2<7(v) | false |  |
|  |  | Fin del mientras | |

Sale con i=2; a[2]=26

### r = (a[i].num == ele)

26==26 (true)

*Implementación del TAD*

#include <iostream>

#include <iomanip>

using namespace std;

class BUSCA{

int num;

public:

void menu(){

cout<< "\nMENU DE OPCIONES\n";

cout<< "----------------\n" ;

cout<<"<1> Ingresar\n";

cout<<"<2> Busqueda\n";

cout<<"<3> Mostrar vector\n";

cout<<"<4> Salir \n";

}

void ingresarDatos(){

fflush(stdin);

cout<<"\n ingresar numero: ";

cin>>num;

}

void mostrarDatos(){

cout<<num<<setw(5);

}

void registrarDatos(BUSCA a[50], int &n, BUSCA x){

a[n]=x;

n++;

}

bool buscaVectorDesordenado (BUSCA a[50] , int n, int ele){

// Definir variables

int i;

bool r;

i = 0;

while (a[i].num != ele && i < n)

i = i + 1;

r = (a[i].num == ele);

return r;

}

void mostrarVector(BUSCA a[50],int num){

// Definir variable

int i;

for(i = 0;i < num;i++)

a[i].mostrarDatos();

}

};

int main(int argc, char\*\* argv) {

char opcion;

BUSCA a[50],x;

int indice=0,ele;

do{

x.menu();

cout<<"\ningrese opcion : ";

opcion=cin.get();

switch(opcion){

case '1':

x.ingresarDatos();

x.registrarDatos(a,indice,x);

break;

case '2':

cout<<"Ingrese elemento a buscar: ";

cin>>ele;

x.buscaVectorDesordenado(a,indice, ele);

if(x.buscaVectorDesordenado(a,indice, ele))

cout<<"Elemento se encuentra en el vector";

else

cout<<"Elemento no se encuentra en el vector";

break;

case '3':

x.mostrarVector(a,indice);

break;

}

cin.ignore();

}

while( opcion !='4');

return 0;

}

### En arreglos ordenados

Cuando el arreglo esta ordenado se impone la condición para controlar la búsqueda, donde

A[0] <= elemento a buscar <= A[N-1]

Si es falso el elemento no esta en el rango.

Sea A el vector

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 8 | 12 | 18 | 26 | 40 |

Elemento a buscar 26

Comparamos cada elemento del vector con el elemento a buscar, como encuentra el elemento a buscar devuelve verdad, ya que 26 pertenece al vector.

**Algoritmo**

Para cada elemento del vector a se determina si es igual al elemento buscado, esta búsqueda es realizado por la función buscaVectorOrdenado, que devuelve verdad si elemento pertenece al vector y falso si elemento no pertenece al vector.

*Especificación del TAD y los algoritmos*

**Especificación** BUSCA

**variable**

num : entero.

**operaciones**

menu :no retorna valor.

ingresarDatos() :no retorna valor.

mostrarDatos() : no retorna valor.

mostrarDatos() : no retorna valor.

registrarDatos(a, n, x) : no retorna valor.

*buscaVectorOrdenado*(a, n, ele) : retorna valor.

mostrarVector(a, n) : no retorna valor.

**significado**

*menu* muestra las opciones a escoger.

*ingresarDatos* ingresa datos al vector.

*mostrarDatos* muestra los datos del vector.

*registrarDatos* registra los datos de *x* en el vector *a.*

*buscaVectorOrdenado* tiene como precondición al vector a, su número de elementos y elemento a buscar, y como postcondición devuelve verdad si elemento pertenece al vector y falso en caso contrario.

*mostrarVector* muestra el vector ordenado.

**Fin\_especificacion**

**Algoritmo**

Funcion buscaVectorOrdenado (a, n, ele): lógico

// Definir variables

Entero: i

Logico: r

i = 0

Mientras (a[i].num < ele and i < n) Hacer

i = i + 1

Fin\_mientras

r = (a[i].num ==ele)

retornar r

Fin\_funcion

**Resultados de la corrida de prueba del algoritmo**

Sea A el vector

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 8 | 12 | 18 | 26 | 40 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |

Elemento a buscar 26

|  |  |  |  |
| --- | --- | --- | --- |
| índice | valores | Resultado de la expresión lógica | Resultados del cuerpo del bucle |
| a[i]!=26 y i<7 | i=i+1 |
| 0 | a[0]=8; 8!=26 (v) y 0<7(v) | true | 1 |
| 1 | a[1]=40; 40!=26 (v) y 1<7(v) | true | 2 |
| 2 | a[2]=26; 26!=26 (f) y 2<7(v) | false |  |
|  |  | Fin del mientras | |

Sale con i=2; a[2]=26

### r = (a[i].num == ele)

26==26 (true)

*Implementación del TAD*

#include <iostream>

#include <iomanip>

using namespace std;

class BUSCA {

int num;

public:

void menu(){

cout<< "\n MENU DE OPCIONES \n";

cout<< "----------------\n" ;

cout<<"<1> Ingresar\n";

cout<<"<2> Busqueda ordenada \n";

cout<<"<3> Mostrar vector \n";

cout<<"<4> Salir \n";

}

void ingresarDatos(){

fflush(stdin);

cout<<"\n leer codigo: ";

cin>>num;

}

void mostrarDatos(){

cout<<num<<setw(5);

}

void registrarDatos(BUSCA a[50], int &n, BUSCA x ){

a[n]=x;

n++;

}

bool buscaVectorOrdenado (BUSCA a[50] , int n, int ele){

// Definir variables

int i;

bool r;

i = 0;

while (a[i].num < ele && i < n)

i = i + 1;

r = (a[i].num ==ele) ;

return r;

}

void mostrarVector(BUSCA a[50],int num){

// Definir variable

int i;

for(i = 0;i < num;i++)

a[i].mostrarDatos();

}

};

int main(int argc, char\*\* argv) {

char opcion;

BUSCA a[50],x;

int indice=0,ele;

do{

x.menu();

cout<<"\n Ingrese opcion : ";

opcion=cin.get();

switch(opcion) {

case '1':

x.ingresarDatos();

x.registrarDatos(a,indice,x);

break;

case '2':

cout<<"Ingrese elemento a buscar: ";

cin>>ele;

x.buscaVectorOrdenado(a,indice, ele);

if(x.buscaVectorOrdenado(a,indice, ele))

cout<<"Elemento se encuentra en el vector";

else

cout<<"Elemento no se encuentra en el vector";

break;

case '3':

x.mostrarVector(a,indice);

break;

}

cin.ignore();

}

while( opcion !='4');

return 0;

}

Resultados de la corrida de prueba del algoritmo

n=7,ele=26

|  |  |  |  |
| --- | --- | --- | --- |
| índice | valores | Resultado de la expresión lógica | Resultados del cuerpo del bucle |
| a[i]<26 y i<7 | i=i+1 |
| 0 | a[0]=1; 1 <26 (v) y 0<7(v) | true | 1 |
| 1 | a[1]=4; 4 <26 (v) y 1<7(v) | true | 2 |
| 2 | a[2]=8; 8 <26 (v) y 2<7(v) | true | 3 |
| 3 | a[3]=12; 12<26 (v) y 3<7(v) | true | 3 |
| 4 | a[4]=18; 18 <26 (v) y 4<7(v) | true | 3 |
| 5 | a[2]=26; 26 <26 (f) y 2<7(v) | fase |  |
|  |  | Fin del mientras | |

Sale con i=5; a[5]=26

### r = (a[i].num == ele)

26==26 (true)

#### Búsqueda binaria

Consiste en dividir el intervalo de búsqueda en dos partes, comparando el elemento buscado con el elemento medio del vector. En caso de que el elemento a buscar sea mayor que el elemento medio, se continúa la búsqueda en la segunda mitad del vector. Si por el contrario el elemento a buscar es menor que el elemento medio, la búsqueda continua en la primera mitad del vector, en ambos casos se redefinen los extremos del intervalo, disminuyendo el espacio de búsqueda, repitiendo el proceso hasta que se encuentre el valor a buscar o hasta que el elemento a buscar no se encuentre en el vector.

Pasos a efectuar para una búsqueda binaria:

* + Hacemos inf = 0 y sup = n1- 1
  + Para Calcular el medio:
  + medio = (inf + sup ) / 2
  + Si estamos en la primera mitad del vector hacemos sup = medio - 1
  + Si estamos en la segunda mitad del vector hacemos inf = medio + 1

Sea A el vector:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 12 | 14 | 18 | 26 | 40 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  |  |  |  |  |
|  |  |  |  | | |  |
| inf |  |  | medio | | | superior |

Elemento a buscar: 26

* + inf = 0 y sup = 6
  + medio = (inf + sup) / 2 = (0 + 6) / 2 = 3  
    a[3] =14 se divide el vector en dos partes

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 4 | 12 | 14 | 18 | 26 | 40 |
| 0 | 1 | 2 | 4 | 5 | 6 |

Como elemento buscado 26 es mayor que elemento central 14, la búsqueda continua en la segunda mitad del vector.

* inf = 4 y sup = 6
* medio = (inf + sup) / 2 = (4 + 6) / 2 = 5, a[5] = 26,
* Como 26 = a[5] la búsqueda acaba, devolviendo verdad, ya que 26 pertenece al vector.

Analizaremos la corrida de prueba para el siguiente vector.-

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 5 | 10 | 15 | 21 | 50 | 60 | 62 | 73 | 81 |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

1. **Elemento existe**

Elemento a buscar = 10

|  |  |  |  |
| --- | --- | --- | --- |
| Límite inferior  inf | Límite superior  sup | Medio valor | Vector medio  a[medio] |
| 0  0  2 | 9  3  3 | 4  1  2 | a[4]=21; 21>10 está en la 1ra mitad  a[1]=5; 5<10 está en la 2da mitad  a[2]=10 encontrado |

1. **Elemento no existe**

Elemento a buscar = 55

|  |  |  |  |
| --- | --- | --- | --- |
| Límite inferior  inf | Límite superior  sup | Medio valor | Vector medio  a[medio] |
| 0  5  5  6 | 9  9  6  6 | 4  7  5  - | a[4]=21; 21<55 está en la 2da mitad  a[7]=62; 62>50 está en la 1ra mitad  a[5]=50; 50<55 está en la 2da mitad  inf<=sup (no encontrado) |

**Algoritmo**

Para cada elemento del vector a se determina si es igual al elemento buscado, esta búsqueda es realizado por la función *binario*, que devuelve verdad si elemento pertenece al vector y falso si elemento no pertenece al vector.

*Especificación del TAD y los algoritmos*

**Especificación** BINARIA

**variable**

num : entero.

**operaciones**

menu : no retorna valor.

ingresarDatos(x) : no retorna valor.

registrarDatos(a,n,x) : no retorna valor.

*buscaBinaria*(a,n,ele) : retorna valor.

visualizarVector(a,n) : no retorna valor.

**significado**

*menu* muestra las opciones a escoger.

*ingresarDatos* ingresa datos al vector.

*mostrarDatos* muestra los datos del vector.

*registrarDatos* registra los datos de *x* en el vector *a.*

*buscaBinaria* tiene como precondición al vector a, su número de elementos y elementos a buscar, y como postcondición devuelve verdad si elemento pertenece al vector y falso en caso contrario.

mostrarVector muestra el vector ordenado.

**Fin\_especificacion**

Funcion buscaBinaria(a, n,ele): lógico

// Declarar e inicializar variables

int inf, sup, medio = 0;

bool recorre = false, p = false;

inf = 0;

sup = n - 1;

Si (ele >= a[0] && ele <= a[n-1]) entonces

recorre = true

else

recorre = false

Fin\_si

Mientras (recorre) Hacer

medio = (inf + sup) / 2;

Si (ele > a[medio]) entonces

inf = medio + 1

Sino

sup = medio- 1

Fin\_si

Si (inf <= sup and a[medio] != ele) entonces

recorre =true

Sino

recorre = false

Fin-si

p = (a[medio].num == ele)

retornar p

Fin\_mientras

Fin\_funcion

***Implementación del TAD***

#include <iostream>

#include <iomanip>

#include <cstdlib>

using namespace std;

class BINARIA{

int num;

public:

void menu(){

cout<< "\nMENU DE OPCIONES\n";

cout<< "----------------\n" ;

cout<<"<1> Ingresar\n";

cout<<"<2> Busqueda binaria\n";

cout<<"<3> Mostrar vector\n";

cout<<"<4> Salir \n";

}

void ingresarDatos(){

fflush(stdin);

cout<<"\n leer codigo: ";

cin>>num;

}

void mostrarDatos(){

cout<<num<<setw(5);

}

void registrarDatos(BINARIA a[50], int &n, BINARIA x ){

a[n]=x;

n++;

}

bool buscaBinaria (BINARIA a[50], int n, int ele) {

// Declarar e inicializar variables

int inf, sup, medio = 0;

bool recorre = false, p = false;

inf = 0;

sup = n - 1;

if (ele >= a[0].num && ele <= a[n-1].num)

recorre = true;

else

recorre = false;

while (recorre) {

medio = (inf + sup) / 2;

if (ele > a[medio].num)

inf = medio + 1;

else

sup = medio- 1;

if (inf <= sup && a[medio].num != ele)

recorre =true;

else

recorre = false;

}

p = (a[medio].num == ele);

return p;

}

void visualizar(BINARIA a[50],int num){

// Definir variable

int i;

for(i = 0;i < num;i++)

a[i].mostrarDatos();

}

};

int main(){

char opcion;

BINARIA a[50],x;

int indice=0,ele;

do{

x.menu();

cout<<"\ningrese opcion : ";

opcion=cin.get();

switch(opcion) {

case '1':

x.ingresarDatos();

x.registrarDatos(a,indice,x);

break;

case '2':

cout<<"Ingrese elemento a buscar: ";

cin>>ele;

x.buscaBinaria(a,indice, ele);

if(x.buscaBinaria(a,indice, ele))

cout<<"Elemento se encuentra en el vector";

else

cout<<"Elemento no se encuentra en el vector";

break;

case '3':

x.visualizar(a,indice);

break;

}

cin.ignore();

}

while( opcion !='4');

system("PAUSE");

return 0;

}