Fundamentos de programación

6 Algoritmos de recorrido y búsqueda para arrays

Grado en Ingeniería Informática
Grado en Ingeniería del Software
Grado en Ingeniería de Computadores

Facultad de Informática
Universidad Complutense

Índice

Luis Hernández Yáñez

Recorrido de arrays

Esquema de recorrido

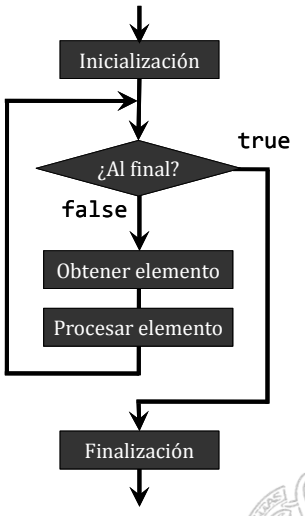
Inicialización

Mientras no se llegue al final de la secuencia:

Obtener el siguiente elemento

Procesar el elemento

Finalización



Secuencias en arrays:

- ✓ Todas las posiciones ocupadas:
  Tamaño del array = longitud de la secuencia
- ✓ Posiciones libres al final del array:
  Tamaño del array > longitud de la secuencia
  - ➢ Con contador de elementos.
  - ➢ Con centinela.

Luis Hernández Yáñez

Recorrido de arrays

Recorrido de secuencias en arrays

- ✓ Todas las posiciones ocupadas:
  N elementos en un array de N posiciones:
  Recorrer todo el array desde la primera posición hasta la última.
- ✓ Posiciones libres al final del array:
  - ➢ Con contador de elementos:
    Recorrer las posiciones del array desde la primera hasta *contador*-1.
  - ➢ Con centinela:
    Recorrer el array hasta encontrar el valor centinela.

Luis Hernández Yáñez

Recorrido de arrays

Recorrido de arrays completos

Todas las posiciones del array ocupadas.

```
const int N = 10;
double ventas[N];
```

ventas

125.40	76.95	328.80	254.62	435.00	164.29	316.05	219.99	93.45	756.62
0	1	2	3	4	5	6	7	8	9

```
int i = 0;
double dato;
while (i < N) {
  dato = ventas[i];
  // Procesar el dato...
  i++;
}
// ...
```

Versión con for

```
double dato;
for (int i = 0; i < N; i++) {
  dato = ventas[i];
  // Procesar el dato...
}
// ...
```

Luis Hernández Yáñez
Pedro J. Martín de la Calle

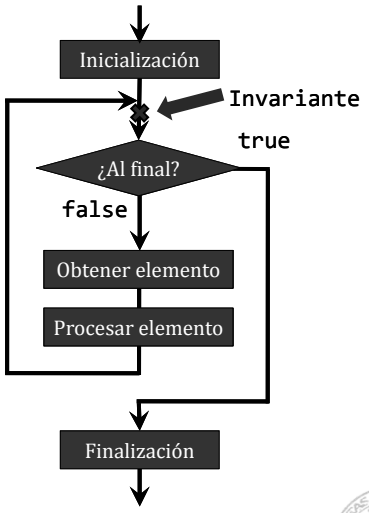
Recorrido de arrays: propiedad invariante

Propiedad invariante

Expresa la relación entre la variables que controlan el bucle

Se cumple vuelta tras vuelta

```
int i = 0;
double dato;
while (i < N) {
  dato = ventas[i];
  // Procesar el dato...
  i++;
}
// ...
```



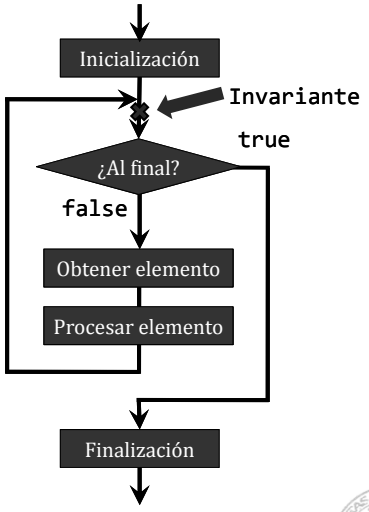
Invariante :

- Procesado(ventas[0 ... i-1])
- 0 ≤ i ≤ N

Luis Hernández Yáñez
Pedro J. Martín de la Calle

Recorrido de arrays: propiedad invariante

```
int i = 0;
double dato = ventas[i]; //N>0
// Procesar el dato...
while (i < (N-1)) {
  i++;
  dato = ventas[i];
  // Procesar el dato...
}
// ...
```



Invariante :

- Procesado(ventas[0 ... i])
- dato= ventas[i]
- 0 ≤ i ≤ N-1

Luis Hernández Yáñez
Pedro J. Martín de la Calle

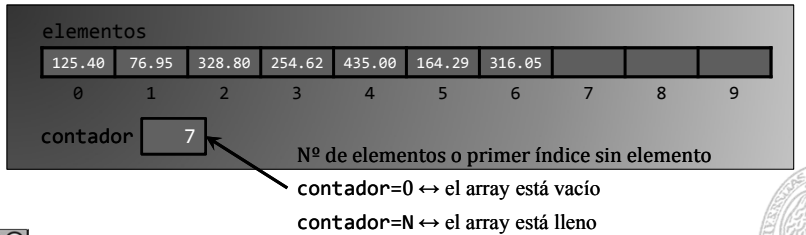
Recorrido de arrays no completos

Recorrido de arrays no completos – con contador

No todas las posiciones del array ocupadas. Contador de elementos.

Como el array y el contador están íntimamente relacionados, usamos una estructura para encapsularlos:

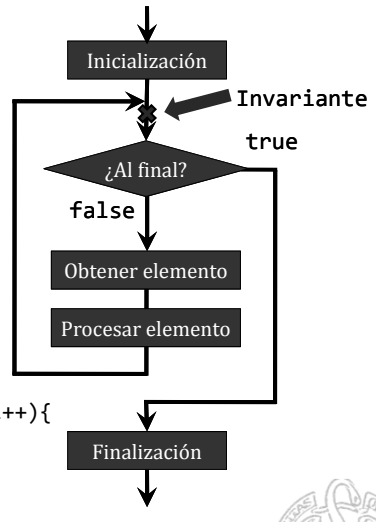
```
const int N = 10;
typedef struct {
  double elementos[N];
  int contador;
} tLista; // struct termina en ;
```



Luis Hernández Yáñez
Pedro J. Martín de la Calle

Recorrido de arrays no completos

```
int i = 0;
double dato;
while (i < miLista.contador) {
  dato = miLista.elementos[i];
  // Procesar el dato...
  i++;
} // ...
```



Invariante :

- Procesado(`miLista.elementos`[0 … `i`-1])
- 0 ≤ `i` ≤ `miLista.contador` ≤ `N`

Versión con `for`

```
double dato;
for (int i = 0; i < miLista.contador; i++){
 dato = miLista.elementos[i];
  // Procesar el dato...
} // ...
```

Luis Hernández Yáñez
Pedro J. Martín de la Calle

Recorrido de arrays no completos

Recorrido de arrays no completos – con centinela

No todas las posiciones del array ocupadas.

```
const int N = 10;
double array[N];
```

Todos los valores positivos: centinela = cualquier negativo

array

0	1	2	3	4	5	6	7	8	9
125.40	76.95	328.80	254.62	435.00	164.29	316.05	-1.0		

```
int i = 0;
double dato;
bool final = false;
while (!final) {
  dato = array[i];
  if (dato < 0) final = true;
  else {
    // Procesar el dato...
    i++;
  }
} // ...
```

Invariante : **esquema asimétrico**

- Procesado(`array`[0 … `i`-1])
- `final` → `array`[`i`] < 0
- `array`[0 … `i`-1] ≥ 0
- 0 ≤ `i` ≤ `N`-1

Luis Hernández Yáñez
Pedro J. Martín de la Calle

Recorrido de arrays no completos

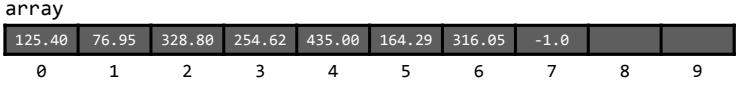
Recorrido de arrays no completos – con centinela

No todas las posiciones del array ocupadas.

```
const int N = 10;
double array[N];
```

Todos los valores positivos: centinela = cualquier negativo

array




```
int i = 0;
double dato = array[i]; // N>0
bool final = dato<0;
while (!final){
  // Procesar el dato...
  i++;
  dato = array[i];
  final = dato<0;
} // …
```

Invariante : **esquema simétrico**

- Procesado(array[0 … i-1])
- final ↔ array[i] < 0
- array[0 … i-1] ≥ 0
- dato = array[i]
- 0 ≤ i ≤ N-1

Luis Hernández Yáñez
Pedro J. Martín de la Calle

Recorrido de arrays no completos

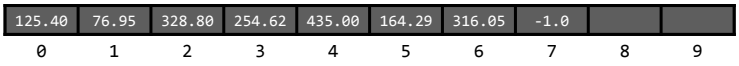
Recorrido de arrays no completos – con centinela

No todas las posiciones del array ocupadas.

```
const int N = 10;
double array[N];
```

Todos los valores positivos: centinela = cualquier negativo

array




```
int i = 0;
double dato = array[i]; // N>0
while (dato >= 0){
  // Procesar el dato...
  i++;
  dato = array[i];
} // …
```

Invariante : **esquema simétrico controlado por expresión centinela**

- Procesado(array[0 … i-1])
- array[0 … i-1] ≥ 0
- dato = array[i]
- 0 ≤ i ≤ N-1

Luis Hernández Yáñez
Pedro J. Martín de la Calle

Ejemplos

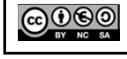
Media de una lista de N números enteros en array media.cpp

```
const int N = 100;
int nums[N]; // Exactamente 100 enteros

// Por ejemplo, los 100 primeros cuadrados:
for (int i = 0; i < N; i++)
  nums[i] = i * i;

double total = 0, media;
for (int i = 0; i < N; i++)
  total += nums[i];
media = total / N;
cout << "Media de la lista de enteros: " << media << endl;
```


Ejemplos

Array con los N primeros números de Fibonacci fibonacci.cpp

```
const int N = 50;
long long int fib[N]; // Exactamente 50 números

fib[0] = 1;
fib[1] = 1;

for (int i = 2; i < N; i++)
  fib[i] = fib[i-1] + fib[i-2];

for (int i = 0; i < N; i++)
  cout << fib[i] << "   ";
```

Ejemplos

Cuenta de valores con k dígitos

Recorrer una lista de N números enteros no negativos contabilizando cuántos son de 1 dígito, de 2 dígitos, de 3 dígitos, etcétera (hasta 6 dígitos).

2 arrays: array con valores y array de contadores

```
const unsigned int N = 100;
unsigned int num[N]; // Exactamente 100 números
const unsigned int D = 6;
unsigned int dig[D];
// Posición i --> cantidad de números de i+1 dígitos
```

Función que devuelve el número de dígitos de un valor entero positivo:

```
unsigned int digitos( unsigned int dato) {
  unsigned int n_digitos = 1;
  while (dato >= 10) {
    dato = dato / 10;
    n_digitos++;
  }
  return n_digitos;
}
```

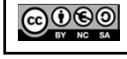

Ejemplos

Cuenta de valores con k dígitos

digitos.cpp

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

unsigned int digitos(unsigned int);

int main() {
  const unsigned int N = 100;
  unsigned int num[N]; // Exactamente 100 números
  const unsigned int D = 6;
  unsigned int dig[D];
  // Posición i --> cantidad de números de i+1 dígitos

  for (unsigned int i = 0; i < N; i++) num[i] = rand();
  for (unsigned int i = 0; i < D; i++) dig[i] = 0;

  int dato;
  for (unsigned int i = 0; i < N; i++) {
    dato = num[i];
    unsigned int pos = digitos(dato) - 1;
    if (pos < D) dig[pos]++;
  }
```

...



Ejemplos

```
...

  for (unsigned int i = 0; i < D; i++)
    cout << "De " << i+1 << " dig. = " << dig[i] << endl;

  return 0;
}

unsigned int digitos(unsigned int dato) {
  unsigned int n_digitos = 1;
  while (dato >= 10) {
    dato = dato / 10;
    n_digitos++;
  }
  return n_digitos;
}
```

Luis Hernández Yáñez

De archivo a array

Carga de los datos de un array desde un archivo

En ocasiones puede resultar conveniente cargar en un array los datos de una secuencia que se encuentre en un archivo. Por ejemplo, si luego hay que realizar varios recorridos.

```
const int N = 100;
typedef struct {
  double elementos[N];
  int contador;
} tLista;

ifstream archivo;
archivo.open("datos.txt");
tLista miLista;
int contador = 0; // hemos leído 0 datos
bool encontrado = (contador == N); // Buscamos el N-ésimo dato del archivo
double dato;
bool eof= !(archivo >> dato);
while (!eof && !encontrado) { // Buscamos el N-ésimo dato del archivo
    miLista.elementos[contador] = dato;
    contador++;
    encontrado = (contador == N);
    eof = !(archivo >> dato);
}
miLista.contador = contador;
```

Hay que crear un array suficientemente grande y asegurarse de que no haya más datos que posiciones en el array.

Luis Hernández Yáñez
Pedro J. Martín de la Calle

De archivo a array

carga.cpp

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

int main() {
  const int N = 100;
  typedef struct {
    double elementos[N];
    int contador;
  } tLista;
  tLista miLista;

  ifstream archivo;
  double dato;

  int contador = 0; // hemos leído 0 datos
  bool encontrado = (contador == N); // buscamos el N-ésimo dato

  archivo.open("datos.txt");
  if (!archivo.is_open())
    cout << "ERROR: No existe el archivo!" << endl;
  else {
```

Luis Hernández Yáñez
Pedro J. Martín de la Calle

De archivo a array

```
// Buscar en archivo el N-ésimo dato
bool eof= !(archivo >> dato);
while (!eof && !encontrado) {
  miLista.elementos[contador] = dato;
  contador++;
  encontrado = (contador == N);
  eof = !(archivo >> dato);
}
miLista.contador = contador;


if(eof && !encontrado) {
   // Había menos de N datos…
}else if(eof && encontrado) {
   // Había exactamente N datos…
}else{ // (!eof && encontrado)
   // Había más de N datos…
}
```

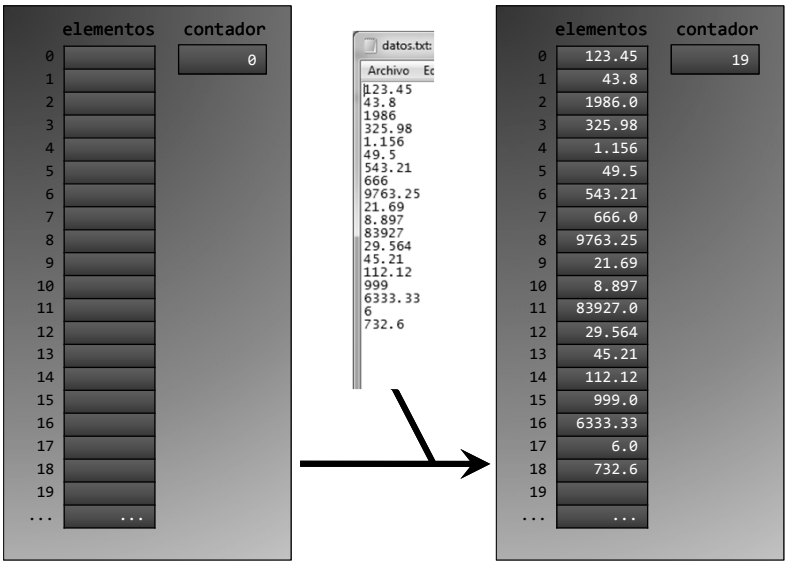
Luis Hernández Yáñez
Pedro J. Martín de la Calle

De archivo a array

```
    // Mostramos los datos cargados
    for (int i = 0; i < miLista.contador; i++) { // Recorremos array
        dato = miLista.elementos[i];
        cout << dato << "   ";
    }
    cout << endl;

    archivo.close();
  } // else

  return 0;
} // main
```

Luis Hernández Yáñez
Pedro J. Martín de la Calle

De archivo a array



Luis Hernández Yáñez

De array a archivo

Copia de los datos de un array a un archivo

A menudo querremos guardar en disco (memoria permanente) los datos de un array, para recuperarlos más adelante.

En principio, el array puede ser todo lo grande que haga falta.

```
const int N = 100;
typedef struct {
  double elementos[N];
  int contador;
} tLista;
tLista miLista;
ofstream archivo;
double dato;
...
archivo.open("backup.txt");
for (int i = 0; i < miLista.contador; i++) // Recorremos el array
  archivo << miLista.elementos[i] << endl;
archivo.close();
```

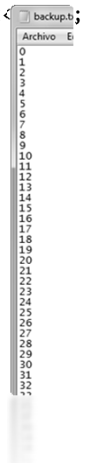

De array a archivo

backup.cpp

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

int main() {
  const int N = 100;
  typedef struct {
    double elementos[N];
    int contador;
  } tLista;
  tLista miLista;

  ofstream archivo;
  double dato;

  for (int i = 0; i < N; i++)
    miLista.elementos[i] = i;

  miLista.contador = N; ...
```


De array a archivo

```
  ...
  archivo.open("backup.txt");

  if (!archivo.is_open())
    cout << "ERROR: No se ha podido crear el archivo!" <<
  else {
    // Ahora creamos la copia de seguridad...
    for (int i = 0; i < miLista.contador; i++)
      archivo << miLista.elementos[i] << endl;
    archivo.close();
  }

  return 0;
}
```



Luis Hernández Yáñez

Arrays multidimensionales

Recorrido de un array bidimensional

```
const int FILAS = 10;
const int COLUMNAS = 5;
typedef double tMatriz[FILAS][COLUMNAS];
tMatriz matriz;
```

Para cada *fila* (de 0 a FILAS – 1):

  Para cada *columna* (de 0 a COLUMNAS – 1):

    Procesar el elemento en [*fila*][*columna*]

```
for (int fila = 0; fila < FILAS; fila++)
  for (int columna = 0; columna < COLUMNAS; columna++)
    // Procesar matriz[fila][columna];
```

Luis Hernández Yáñez

Arrays multidimensionales

Recorrido de arrays N-dimensionales

```
const int DIM1 = 10;
const int DIM2 = 5;
const int DIM3 = 25;
const int DIM4 = 50;
typedef double tMatriz[DIM1][DIM2][DIM3][DIM4];
tMatriz matriz;
```

Anidamiento de bucles desde la primera dimensión hasta la última:

```
for (int n1 = 0; n1 < DIM1; n1++)
  for (int n2 = 0; n2 < DIM2; n2++)
    for (int n3 = 0; n3 < DIM3; n3++)
      for (int n4 = 0; n4 < DIM4; n4++)
        // Procesar matriz[n1][n2][n3][n4];
```


Arrays multidimensionales

Ventas diarias de cuatro sucursales

Para cada mes del año: ingresos de cada sucursal cada día del mes.

Meses con distinto nº de días → junto con la matriz de ventas mensual guardamos el nº de días del mes concreto → estructura.

```
const int MESES = 12;
const int DIAS = 31;
const int SUCURSALES = 4;
typedef double tVentaMes[DIAS][SUCURSALES];
typedef struct {
  tVentaMes ventas;
  int dias;
} tMes;
typedef tMes tVentaAnual[MESES];
tVentaAnual anual;
```

```
anual → tVentaAnual
anual[i] → tMes
anual[i].dias → int
anual[i].ventas → tVentaMes
anual[i].ventas[j][k] → double
```


Arrays multidimensionales

Ventas diarias de cuatro sucursales

```
typedef double tVentaMes[DIAS][SUCURSALES];
typedef struct {
  tVentaMes ventas;
  int dias;
} tMes;
typedef tMes tVentaAnual[MESES];
tVentaAnual anual;
```

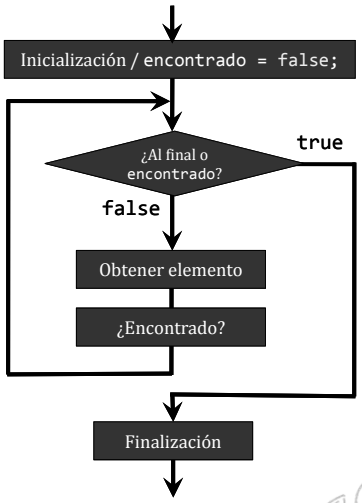
Cálculo del total de las ventas del año:

```
double total = 0;
for (int mes = 0; mes < MESES; mes++)
  for (int dia = 0; dia < anual[mes].dias; dia++)
    for (int suc = 0; suc < SUCURSALES; suc++)
      total += anual[mes].ventas[dia][suc];
```


Búsquedas en arrays

Esquema de búsqueda

- Inicialización
- Mientras no se encuentre el elemento y no se esté al final de la secuencia:
  - Obtener el siguiente elemento
  - Comprobar si el elemento satisface la condición
- Finalización (tratar el elemento encontrado o indicar que no se ha encontrado)



Búsquedas en arrays

Búsquedas en arrays completos

Todas las posiciones del array ocupadas:

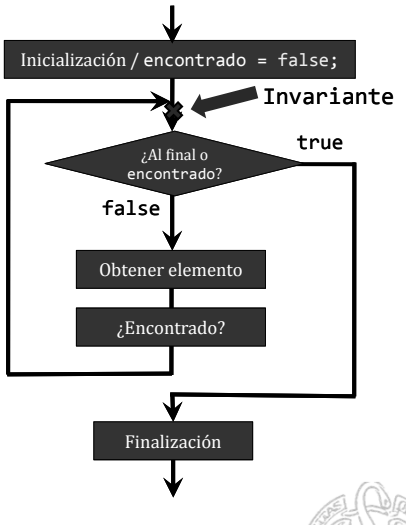
```
const int N = 100;
int array[N];
int buscado;

// ...
cout << "Introduce el valor a buscar (entero): ";
cin >> buscado;
int pos = 0;
bool encontrado = false;
while ((pos < N) && !encontrado){
// Mientras no se llegue al final del array y no encontrado
  if (array[pos] == buscado) encontrado = true;
  else pos++;
}
if (encontrado) // ...
```

Luis Hernández Yáñez

Búsquedas en arrays: invariantes

Invariante: ***esquema asimétrico***

- `encontrado` → (`array[pos]=buscado`))
- `buscado` no está en `array[0..pos-1]`
- 0 ≤ `pos` ≤ `N`



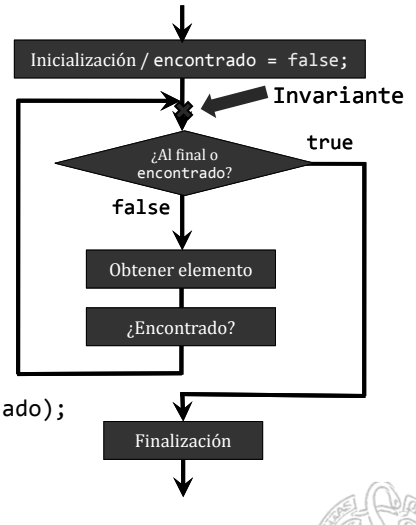

```
cin >> buscado;
int pos = 0;
bool encontrado = false;
while ((pos < N) && !encontrado){
  if (array[pos] == buscado)
     encontrado = true;
  else pos++;
}
if (encontrado) // ...
```

Luis Hernández Yáñez
Pedro J. Martín de la Calle

Búsquedas en arrays: invariantes

Invariante: ***esquema simétrico 1***

- encontrado ↔ (array[pos]=buscado)), salvo cuando pos = -1
- buscado no está en array[0..pos-1]
- -1 ≤ pos ≤ N-1



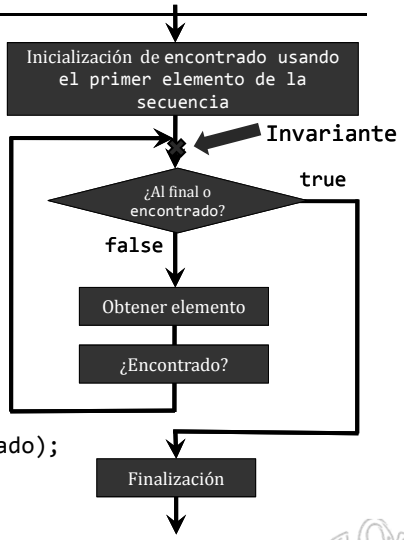

```
cin >> buscado;
int pos = -1;
bool encontrado = false;
while((pos<(N-1)) && !encontrado){
  pos++;
  encontrado = (array[pos] == buscado);
}
if (encontrado) // ...
```

Luis Hernández Yáñez
Pedro J. Martín de la Calle

Búsquedas en arrays: invariantes

Invariante: ***esquema simétrico 2***

- encontrado ↔ (array[pos]=buscado))
- buscado no está en array[0..pos-1]
- 0 ≤ pos ≤ N-1




```
cin >> buscado;
int pos = 0;
bool encontrado =
  (array[pos] == buscado); //N>0
while((pos<(N-1)) && !encontrado){
  pos++;
  encontrado = (array[pos] == buscado);
}
if (encontrado) // ...
```

Luis Hernández Yáñez
Pedro J. Martín de la Calle

Búsquedas en arrays

Búsquedas en arrays no completos – con contador

No todas las posiciones del array ocupadas. Contador de elementos.

```
const int N = 10;
typedef struct {
  double elementos[N];
  int contador;
} tLista; // struct termina en ;
tLista miLista;
int buscado;

// ESQUEMA ASIMÉTRICO...
cout << "Introduce el valor a buscar (entero): ";
cin >> buscado;
int pos = 0;
bool encontrado = false;
while ((pos < miLista.contador) && !encontrado)
// Mientras no se llegue al final de la lista y no encontrado
  if (miLista.elementos[pos] == buscado) encontrado = true;
  else pos++;
if (encontrado) // ...
```

Invariante: **esquema asimétrico**

- encontrado → (miLista.elementos[pos]=buscado)
- 0 ≤ pos ≤ miLista.contador
- buscado no está en miLista.elementos[0..pos-1]

Luis Hernández Yáñez
Pedro J. Martín de la Calle

Búsquedas en arrays

Búsquedas en arrays no completos – con contador

No todas las posiciones del array ocupadas. Contador de elementos.

```
const int N = 10;
typedef struct {
  double elementos[N];
  int contador;
} tLista; // struct termina en ;
tLista miLista;
int buscado;

// ESQUEMA SIMÉTRICO 1...
cout << "Introduce el valor a buscar (entero): ";
cin >> buscado;
int pos = -1;
bool encontrado = false;
while ((pos < (miLista.contador-1)) && !encontrado) {
   pos++;
   encontrado = (miLista.elementos[pos] == buscado);
}
if (encontrado) // ...
```

Invariante: **esquema simétrico 1**

- encontrado ↔ (miLista.elementos[pos]=buscado), salvo cuando pos = -1
- -1 ≤ pos ≤ miLista.contador-1
- buscado no está en miLista.elementos[0..pos-1]

Luis Hernández Yáñez
Pedro J. Martín de la Calle

Búsquedas en arrays

Búsquedas en arrays no completos – con centinela

No todas las posiciones del array ocupadas.

Array de valores enteros positivos: centinela = -1

```
const int N = 10;
int array[N];
int centinela = -1;
int buscado;


// ESQUEMA ASIMÉTRICO...
cout << "Introduce el valor a buscar (entero): ";
cin >> buscado;
int pos = 0;
bool final = false, encontrado = false;
while (!final && !encontrado){
// Mientras no se llegue al centinela y no encontrado
  if (array[pos] == centinela) final = true;
  else if (array[pos] == buscado) encontrado = true;
  else pos++;
}
if (encontrado) // ...
```

Invariante: **esquema asimétrico**

- final → (array[pos] = centinela)
- encontrado → (array[pos] = buscado)
- centinela no está en array[0..pos-1]
- buscado no está en array[0..pos-1]
- 0 ≤ pos ≤ N-1

Luis Hernández Yáñez
Pedro J. Martín de la Calle

Búsquedas en arrays

Búsquedas en arrays no completos – con centinela

No todas las posiciones del array ocupadas.

Array de valores enteros positivos: centinela = -1

```
const int N = 10;
int array[N];
int centinela = -1, buscado;


// ESQUEMA SIMÉTRICO 2
cout << "Introduce el valor a buscar (entero): ";
cin >> buscado;
int pos = 0;
bool final = (array[pos] == centinela);    // N>0
bool encontrado = (array[pos] == buscado); // N>0
while (!final && !encontrado) {
   pos++;
   final = (array[pos] == centinela);
   encontrado = (array[pos] == buscado);
}
if (encontrado && !final) // …
```

Invariante: **esquema simétrico 2**

- final ↔ (array[pos] = centinela)
- encontrado ↔ (array[pos] = buscado)
- centinela no está en array[0..pos-1]
- buscado no está en array[0..pos-1]
- 0 ≤ pos ≤ N-1

Luis Hernández Yáñez
Pedro J. Martín de la Calle

Ejemplos

Primer valor por encima de un umbral

umbral.cpp

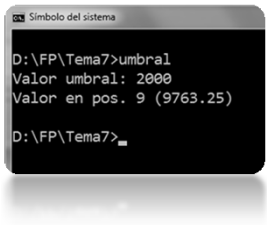
```
#include <iostream>
#include <fstream>
using namespace std;

const int N = 100;
typedef struct {
  double elementos[N];
  int contador;
} tLista;

bool cargar(tLista& lista);

int main() {
  tLista lista;
  if (!cargar(lista))
    cout << "Error de archivo: inexistente o con demasiados datos"
         << endl;
                                                                    ...
```


Ejemplos

```
  else {
    // ESQUEMA ASIMÉTRICO...
    double umbral;
    cout << "Valor umbral: "; cin >> umbral;
    bool encontrado = false;
    int pos = 0;
    while ((pos < lista.contador) && !encontrado){
      if (lista.elementos[pos] > umbral) encontrado = true;
      else pos++;
    }
    if (encontrado)
      cout << "Valor en pos. " << pos + 1 << " ("
           << lista.elementos[pos] << ")" << endl;
    else cout << "No encontrado!" << endl;
  }

  return 0;
}                                                                   ...
```


Ejemplos

```
bool cargar(tLista& lista) {
  ifstream archivo;
  bool abierto = true;
  bool encontrado;
  archivo.open("datos.txt");
  if (!archivo.is_open())
    abierto = false;
  else {
    // Buscar en archivo el N-ésimo dato
    double dato;
    bool eof= !(archivo >> dato);
    int contador = 0; // hemos leído 0 datos
    encontrado = (contador == N); // buscamos el N-ésimo dato
    while (!eof && !encontrado) {
      miLista.elementos[contador] = dato;
      contador++;
      encontrado = (contador == N);
      eof = !(archivo >> dato);
    }
    miLista.contador = contador;
  }
  return abierto && !(encontrado && !eof); //abierto && eof
}
```



Luis Hernández Yáñez
Pedro J. Martín de la Calle

Búsquedas en arrays

Búsquedas por posición

Acceso directo: *array*[*posición*]

Si se puede calcular la posición del elemento, su acceso será directo.

```
typedef double tVentaMes[DIAS][SUCURSALES];
typedef struct {
  tVentaMes ventas;
  int dias;
} tMes;
typedef tMes tVentaAnual[MESES];
tVentaAnual anual;
```

Localizar las ventas del cuarto día del tercer mes en la primera sucursal:

```
anual[2].ventas[3][0]
```

Luis Hernández Yáñez

Búsquedas en arrays multidimensionales

- *Se trata de buscar en cada dimensión*
- *Estas búsquedas deben anidarse usando subprogramas*
- *Ejemplo:* Primer valor por encima de un umbral

```
tVentaAnual anual;
double umbral;
cout << "Valor umbral: ";
cin >> umbral;

// ESQUEMA ASIMÉTRICO...
bool encontrado = false;
int mes = 0, dia, suc;
while ((mes < MESES) && !encontrado){
  buscarDia( anual, umbral, mes, dia, suc, encontrado);
  if (!encontrado) mes++;
}
if (encontrado) ...
```

```
typedef double tVentaMes[DIAS][SUCURSALES];
typedef struct {
  tVentaMes ventas;
  int dias;
} tMes;
typedef tMes tVentaAnual[MESES];
tVentaAnual anual;
```

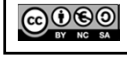
Luis Hernández Yáñez
Pedro J. Martín de la Calle

Búsquedas en arrays multidimensionales

```
void buscarDia( tVentaAnual anual, double umbral, int mes, //entrada
                int& dia, int& suc, bool& encontrado){ // salida

  // ESQUEMA SIMÉTRICO 1...
  dia = -1;
  encontrado = false;
  while ((dia < (anual[mes].dias-1)) && !encontrado) {
     dia++;
     buscarSucursal( anual, umbral, mes, dia, suc, encontrado);
  }
}
```

```
typedef double tVentaMes[DIAS][SUCURSALES];
typedef struct {
  tVentaMes ventas;
  int dias;
} tMes;
typedef tMes tVentaAnual[MESES];
tVentaAnual anual;
```

Luis Hernández Yáñez
Pedro J. Martín de la Calle



Búsquedas en arrays multidimensionales

```
void buscarSucursal( tVentaAnual anual, double umbral, // entrada
                     int mes, int dia, // entrada
                     int& suc, bool& encontrado){ // salida

  // ESQUEMA SIMÉTRICO 2...
  // SUCURSALES > 0
  suc = 0;
  encontrado = (anual[mes].ventas[dia][suc] > umbral);
  while ((suc < (SUCURSALES-1)) && !encontrado) {
     suc++;
     encontrado = (anual[mes].ventas[dia][suc] > umbral);
  }
}
```

```
typedef double tVentaMes[DIAS][SUCURSALES];
typedef struct {
  tVentaMes ventas;
  int dias;
} tMes;
typedef tMes tVentaAnual[MESES];
tVentaAnual anual;
```

Luis Hernández Yáñez
Pedro J. Martín de la Calle

Cadenas de caracteres

Recorridos y búsquedas en cadenas de caracteres

Longitud de la cadena:

- `size()` y `length()` para `string`
- `strlen()` para `char*`

Caso similar a los arrays con contador de elementos.

Recorrido de una cadena generando otra con los caracteres al revés:

inversa.cpp

```
string cadena, inversa;
// ...
int pos = 0;
while (pos < cadena.size()) {
// Mientras no se llegue al final de la cadena
  char car = cadena.at(pos);
  inversa = car + inversa;
  // Se puede concatenar un carácter a una cadena
  pos++;
}
// ...
```

Luis Hernández Yáñez

Cadenas de caracteres

Recorridos y búsquedas en cadenas de caracteres

Búsqueda de un carácter en una cadena:

busca.cpp

```
string cadena;
char buscado;
// ...
cout << "Introduce el carácter a buscar: ";
cin >> buscado;

// ESQUEMA ASIMÉTRICO...
int pos = 0;
bool encontrado = false;
while ((pos < cadena.size()) && !encontrado) {
// Mientras no se llegue al final de la cadena y no encontrado
  if (cadena.at(pos) == buscado) encontrado = true;
  else pos++;
}
if (encontrado) // ...
```


Referencias bibliográficas

- *Programación en C++ para ingenieros*
  F. Xhafa et al. Thomson, 2006
- *El lenguaje de programación C++* (Edición especial)
  B. Stroustrup. Addison-Wesley, 2002
- *C++: An Introduction to Computing* (2ª edición)
  J. Adams, S. Leestma, L. Nyhoff. Prentice Hall, 1998

Luis Hernández Yáñez
Pedro J. Martín de la Calle

Acerca de *Creative Commons*

Licencia CC (Creative Commons)

Este tipo de licencias ofrecen algunos derechos a terceras personas bajo ciertas condiciones.

Este documento tiene establecidas las siguientes:

- Reconocimiento (*Attribution*):
  En cualquier explotación de la obra autorizada por la licencia hará falta reconocer la autoría.
- No comercial (*Non commercial*):
  La explotación de la obra queda limitada a usos no comerciales.
- Compartir igual (*Share alike*):
  La explotación autorizada incluye la creación de obras derivadas siempre que mantengan la misma licencia al ser divulgadas.

Pulsa en la imagen de arriba a la derecha para saber más.

Luis Hernández Yáñez