



ANEXO II

Ejemplos de secuencias

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

Luis Hernández Yáñez
Facultad de Informática
Universidad Complutense



Índice

Recorridos	404
Un aparcamiento	405
¿Paréntesis bien emparejados?	409
¿Dos secuencias iguales?	412
Números primos menores que N	413
Búsquedas	417
Búsqueda de un número en un archivo	419
Búsquedas en secuencias ordenadas	420



Recorridos

Luis Hernández Yáñez



Un aparcamiento

Secuencia de caracteres E y S en archivo
E = Entra un coche; S = Sale un coche
¿Cuántos coches quedan al final de la jornada?
Varios casos, cada uno en una línea y terminado en punto
Final: línea sólo con punto

```
parking.txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEESSSS.
EEEESSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSSS.
EEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEEESSSS.
.
```

Luis Hernández Yáñez



Un aparcamiento

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

int main() {
    int coches;
    char c;
    bool terminar = false;
    ifstream archivo;
    archivo.open("parking.txt");
    if (!archivo.is_open()) {
        cout << "¡No se ha podido abrir el archivo!" << endl;
    }
    else {
        // Recorrido...
        archivo.close();
    }
    return 0;
}
```

Luis Hernández Yáñez



Un aparcamiento (recorrido)

```
while (!terminar) {
    archivo >> c;
    if (c == '.') { // . como primer carácter? (centinela)
        terminar = true;
    }
    else {
        coches = 0;
        while (c != '.') { // Recorrido de la secuencia
            cout << c;
            if (c == 'E') {
                coches++;
            }
            else if (c == 'S') {
                coches--;
            }
            archivo >> c;
        }
        ...
    }
}
```

Luis Hernández Yáñez



```
if (coches >= 0) {
    cout << endl << "Quedan " << coches << " coches.";
}
else {
    cout << endl << "Error: Más salidas que entradas!";
}
cout << endl;
}
```

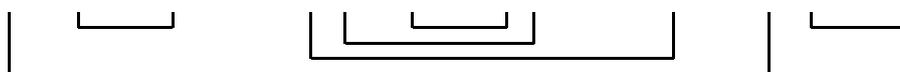


¿Paréntesis bien emparejados?

Cada paréntesis, con su pareja

Secuencia de caracteres terminada en # y con parejas de paréntesis:

a b (c (d e) f g h ((i (j k)) l m n) o p) (r s) #



Contador del nivel de anidamiento:

Al encontrar '(' incrementamos – Al encontrar ')' decrementamos

Al terminar, el contador deberá tener el valor 0

Errores:

– Contador -1: paréntesis de cierre sin uno de apertura pendiente
abc)de(fgh(ij))#

– Contador termina con un valor positivo
Más paréntesis de apertura que de cierre

Algún paréntesis sin cerrar: (a(b(cd(e)f)gh(i)))jk#



¿Paréntesis bien emparejados?

Un error puede interrumpir el recorrido:

```
char c;
int anidamiento = 0, pos = 0;
bool error = false;
cin >> c;
while ((c != '#') && !error) {
    pos++;
    if (c == '(') {
        anidamiento++;
    }
    else if (c == ')') {
        anidamiento--;
    }
    if (anidamiento < 0) {
        error = true;
    }
    if (!error) {
        cin >> c;
    }
}
```

Luis Hernández Yáñez



¿Paréntesis bien emparejados?

parenthesis.cpp

```
if (error) {
    cout << "Error: cierre sin apertura (pos. " << pos
        << ")";
}
else if (anidamiento > 0) {
    cout << "Error: Apertura sin cierre";
}
else {
    cout << "Correcto";
}
cout << endl;
```

```
D:\Docencia\FP\2013-2014\Lessons\Less03\Examples\ESP>parenthesis
ab(c(de) fgh((i(jk))lmn)op)(rs)#
Correcto

D:\Docencia\FP\2013-2014\Lessons\Less03\Examples\ESP>parenthesis
)ab(((cd)ef))#
Error: cierre sin apertura (pos. 1)

D:\Docencia\FP\2013-2014\Lessons\Less03\Examples\ESP>parenthesis
(((abc(d)e(fg(h))))#
Error: Apertura sin cierre
```

Luis Hernández Yáñez



¿Dos secuencias iguales?

iguales.cpp

```
bool iguales() {
    bool sonIguales = true;
    double d1, d2;
    ifstream sec1, sec2;
    bool final = false;
    sec1.open("secuencia1.txt");
    sec2.open("secuencia2.txt");
    sec1 >> d1;
    sec2 >> d2; // Al menos estarán los centinelas (0)
    while (sonIguales && !final) {
        sonIguales = (d1 == d2);
        final = ((d1 == 0) || (d2 == 0));
        if (!final) {
            sec1 >> d1;
            sec2 >> d2;
        }
    }
    sec1.close();
    sec2.close();
    return sonIguales;
}
```



Cambia secuencia2.txt por secuencia3.txt y por secuencia4.txt para comprobar otros casos



Números primos menores que N

primos.cpp

Secuencia calculada: números divisibles sólo por 1 y ellos mismos ($< N$)

```
#include <iostream>
using namespace std;
bool primo(int n);
int main() {
    int num, candidato;
    cout << "Entero en el que parar (>1): ";
    cin >> num;
    if (num > 1) {
        candidato = 2; // El 1 no se considera un número primo
        while (candidato < num) {
            cout << candidato << " "; // Mostrar número primo
            candidato++;
            while (!primo(candidato)) { // Siguiendo primo
                candidato++;
            }
        }
    }
    return 0;
}
```



Números primos menores que N

```
bool primo(int n) {
    bool esPrimo = true;

    for (int i = 2; i <= n - 1; i++) {
        if (n % i == 0) {
            esPrimo = false; // Es divisible por i
        }
    }

    return esPrimo;
}
```



Números primos menores que N

primos2.cpp

Mejoras: probar sólo impares; sólo pueden ser divisibles por impares; no pueden ser divisibles por ninguno mayor que su mitad

```
    candidato = 2;
    cout << candidato << " "; // Mostrar el número primo 2
    candidato++; // Seguimos con el 3, que es primo
    while (candidato < num) {
        cout << candidato << " "; // Mostrar número primo
        candidato = candidato + 2; // Sólo probamos impares
        while (!primo(candidato)) { // Siguiendo número primo
            candidato = candidato + 2;
        }
    } ...

bool primo(int n) {
    bool esPrimo = true;
    for (int i = 3; i <= n / 2; i = i + 2) {
        if (n % i == 0) {
            esPrimo = false; // Es divisible por i
        }
    }
    ...
}
```



Otra mejora más: Paramos al encontrar el primer divisor

```
bool primo(int n) {
    bool esPrimo = true;

    int i = 3;
    while ((i <= n / 2) && esPrimo) {
        if (n % i == 0) {
            esPrimo = false;
        }
        i = i + 2;
    }

    return esPrimo;
}
```



Fundamentos de la programación

Búsquedas



Búsqueda de un número en un archivo

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

int busca(int n);
// Devuelve la línea en la que se encuentra o -1 si no está

int main() {
    int num, linea;

    cout << "Valor a localizar: ";
    cin >> num;
    linea = busca(num);
    if (linea != -1) {
        cout << "Encontrado (línea " << linea << ")" << endl;
    }
    else {
        cout << "No encontrado" << endl;
    }
    return 0;
}
```

buscaarch.cpp

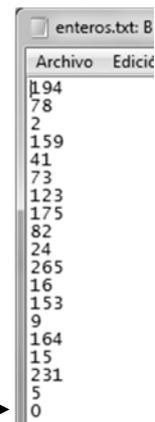
Luis Hernández Yáñez



Búsqueda de un número en un archivo

```
int busca(int n) {
    int i, linea = 0;
    bool encontrado = false;
    ifstream archivo;
    archivo.open("enteros.txt");
    if (!archivo.is_open()) {
        linea = -1;
    }
    else {
        archivo >> i;
        while ((i != 0) && !encontrado) {
            linea++;
            if (i == n) {
                encontrado = true;
            }
            archivo >> i;
        }
        if (!encontrado) {
            linea = -1;
        }
        archivo.close();
    }
    return linea;
}
```

Centinela →



Luis Hernández Yáñez



Búsquedas en secuencias ordenadas

Luis Hernández Yáñez



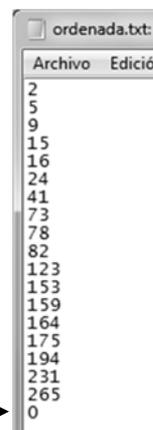
Búsqueda en secuencias ordenadas

Secuencia ordenada de menor a mayor:
paramos al encontrar uno mayor o igual al buscado

`buscaord.cpp`

Los que resten serán seguro mayores: *¡no puede estar el buscado!*

```
cout << "Valor a localizar: ";  
cin >> num;  
archivo >> i;  
while ((i != 0) && (i < num)) {  
    cont++;  
    archivo >> i;  
}  
if (i == num) {  
    cout << "Encontrado (pos.: " << cont << ")";  
}  
else {  
    cout << "No encontrado";  
}  
cout << endl;  
archivo.close();
```



Luis Hernández Yáñez



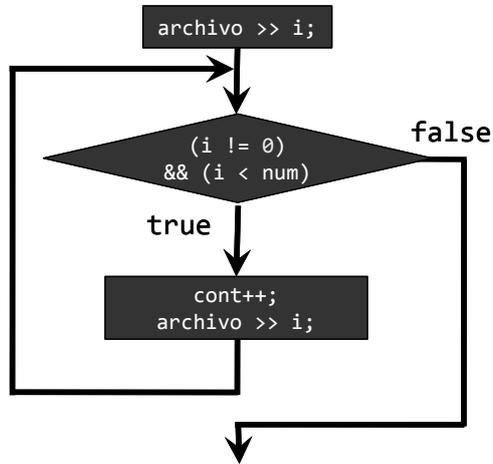
Secuencias ordenadas

Si el elemento está: procesamiento similar a secuencias desordenadas

↓
2 5 9 15 16 24 41 73 78 82 123 153 159 ...

num 9

i 9



Luis Hernández Yáñez



Secuencias ordenadas

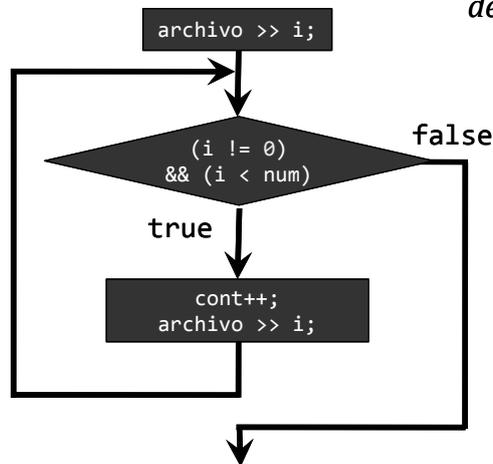
Si el elemento no está: evitamos buscar en el resto de la secuencia

↓
2 5 9 15 16 24 41 73 78 82 123 153 159 ...

No se procesa el resto de la secuencia

num 10

i 15



Luis Hernández Yáñez





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.

