



ANEXO

Más sobre subprogramas

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

Luis Hernández Yáñez
Facultad de Informática
Universidad Complutense



Índice

Archivos como parámetros	498
La función <code>main()</code>	501
Argumentos implícitos	504
Sobrecarga de subprogramas	508



Fundamentos de la programación

Archivos como parámetros



Archivos como parámetros

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

void sumatorio_archivo(ifstream &arch, double &suma);

int main() {
    double resultado;
    ifstream archivo;
    archivo.open("datos.txt");
    if (!archivo.is_open()) {
        cout << "ERROR DE APERTURA" << endl;
    }
    else {
        sumatorio_archivo(archivo, resultado)
        cout << "Suma = " << resultado << endl;
        archivo.close();
    }

    return 0;
}
```



Archivos como parámetros

```
void sumatorio_archivo(ifstream &arch, double &suma) {  
    double dato;  
  
    suma = 0;  
    arch >> dato;  
  
    while (dato != -1) {  
        suma = suma + dato;  
        arch >> dato;  
    }  
}
```



Los archivos siempre se pasan por referencia



Fundamentos de la programación

La función `main()`



Parámetros de `main()`

Comunicación con el sistema operativo

Parámetros opcionales de la función `main()`:

```
int main(int argc, char *argv[])
```

Para obtener datos proporcionados al ejecutar el programa:

```
C:\>prueba cad1 cad2 cad3
```

Ejecuta `prueba.exe` con tres argumentos (cadenas)

Parámetros de `main()`:

- `argc`: número de argumentos que se proporcionan
4 en el ejemplo (primero: nombre del programa con su ruta)
- `argv`: array con las cadenas proporcionadas como argumentos



Lo que devuelve main()

¿Cómo ha ido la función?

La función `main()` devuelve al S.O. un código de terminación

- `0`: *Todo OK*
- Distinto de `0`: *¡Ha habido un error!*

Si la ejecución llega al final de la función `main()`, todo OK:

...

```
return 0; // Fin del programa  
}
```



Fundamentos de la programación

Argumentos implícitos



Argumentos implícitos

Valores predeterminados para parámetros por valor

Valor por defecto para un parámetro:

Tras un = a continuación del nombre del parámetro:

```
void proc(int i = 1);
```

Si no se proporciona argumento, el parámetro toma ese valor

proc(12); i toma el valor explícito **12**

proc(); i toma el valor implícito (**1**)

Sólo puede haber argumentos implícitos en los parámetros finales:

```
void p(int i, int j = 2, int k = 3); // CORRECTO
```

```
void p(int i = 1, int j, int k = 3); // INCORRECTO
```



Una vez asignado un valor implícito, todos los que siguen han de tener también valor implícito



Argumentos implícitos

Parámetros y argumentos implícitos

```
void p(int i, int j = 2, int k = 3);
```

Se copian los argumentos en los parámetros del primero al último
→ los que no tengan correspondencia tomarán los implícitos

```
void p(int i, int j = 2, int k = 3);
```

...

```
p(13); // i toma 13, j y k sus valores implícitos
```

```
p(5, 7); // i toma 5, j toma 7 y k su valor implícito
```

```
p(3, 9, 12); // i toma 3, j toma 9 y k toma 12
```



Los argumentos implícitos se declaran en el prototipo (preferible) o en la cabecera del subprograma, pero NO en ambos



Ejemplo

Por defecto, signo +
Por defecto, Δ es 1

$$f(x, y) = \pm\Delta \frac{x}{y}$$

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

```
double f(double x, double y, int signo = 1, double delta = 1.0);
```

```
int main() {
    double x, y;
    cout << "X = ";
    cin >> x;
    cout << "Y = ";
    cin >> y;
    cout << "signo y delta por defecto: " << f(x, y) << endl;
    cout << "signo -1 y delta por defecto: " << f(x, y, -1) << endl;
    cout << "signo y delta concretos: " << f(x, y, 1, 1.25) << endl;

    return 0;
}
```



No podemos dejar signo por defecto
y concretar delta

```
double f(double x, double y, int signo, double delta) {
    return signo * delta * x / y;
}
```



Fundamentos de la programación

Sobrecarga de subprogramas



Sobrecarga de subprogramas

Igual nombre, distintos parámetros

Funciones o procedimientos con igual nombre y distintos parámetros:

```
int abs(int n);
```

```
double abs(double n);
```

```
long int abs(long int n);
```

Se ejecutará la función que corresponda al tipo de argumento:

```
abs(13) // argumento int --> primera función
```

```
abs(-2.3) // argumento double --> segunda función
```

```
abs(3L) // argumento long int --> tercera función
```



Para indicar que es un literal **long int**, en lugar de **int**



Sobrecarga de subprogramas

inter.cpp

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

void intercambia(int &x, int &y);
void intercambia(double &x,
                  double &y);
void intercambia(char &x, char &y);

void intercambia(int &x, int &y) {
    int tmp;
    tmp = x;
    x = y;
    y = tmp;
}

void intercambia(double &x,
                  double &y) {
    double tmp;
    tmp = x;
    x = y;
    y = tmp;
}
```

```
void intercambia(char &x, char &y) {
    char tmp;
    tmp = x;
    x = y;
    y = tmp;
}

int main() {
    int i1 = 3, i2 = 7;
    double d1 = 12.5, d2 = 35.9;
    char c1 = 'a', c2 = 'b';
    cout << i1 << " - " << i2 << endl;
    cout << d1 << " - " << d2 << endl;
    cout << c1 << " - " << c2 << endl;
    intercambia(i1, i2);
    intercambia(d1, d2);
    intercambia(c1, c2);
    cout << i1 << " - " << i2 << endl;
    cout << d1 << " - " << d2 << endl;
    cout << c1 << " - " << c2 << endl;
    return 0;
}
```



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.

