Programación 1

Tema 5

Funciones

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Índice

- **Funciones**
- Estructuración de programas
  - Metodología de programación descendente
- Ámbito y vida
- Comunicación entre funciones

Funciones

- Abstracción de un conjunto de instrucciones
  - a las que se la un nombre determinado
  - para ser invocadas desde algún otro punto del programa
- Sintaxis:
  - Declaración
  - Definición
  - Invocación

Más adelante en el curso, no ahora

Fuente: «Functions». *Cplusplus.com*. 2000–2017. http://www.cplusplus.com/doc/tutorial/functions/ (consultado el 31-10-2018)

Funciones. Sintaxis

```
<definición-función> ::=
    <tipo> <identificador>
        “(” [<lista-parámetros>] “)”
    <bloqueSecuencial>
<lista-parámetros> ::=
    <parámetro> { “,” <parámetro> }
<parámetro> ::=
    <tipo> <identificador>
<bloqueSecuencial> ::=
    “{” { <instrucción> } “}”
```

Funciones. Sintaxis

```
<invocación-función> ::=
    <identificador>
    “(” [ <lista-argumentos> ] “)”
<lista-argumentos> ::=
    <argumento> {“,” <argumento>}
<argumento> ::= <expresión>
```

Funciones. Sintaxis

- Restricciones a la sintaxis:
  - Si el tipo devuelto es distinto de **`void`**, el cuerpo de la función debe devolver un dato del tipo adecuado a través de la instrucción **`return`**.
  - El identificador de la invocación es el mismo que el de la definición.
  - La lista de parámetros (definición) y la de argumentos (invocación) tienen el mismo número de elementos.
  - El tipo del *i*-ésimo argumento en la lista de argumentos es el mismo (o es compatible) con el *i*-ésimo parámetro de la definición.

Funciones. Ejemplo

	a	b	c	d	e	f	g	h
8	57	58	59	60	61	62	63	64
7	49	50	51	52	53	54	55	56
6	41	42	43	44	45	46	47	48
5	33	34	35	36	37	38	39	40
4	25	26	27	28	29	30	31	32
3	17	18	19	20	21	22	23	24
2	9	10	11	12	13	14	15	16
1	1	2	3	4	5	6	7	8

← columnas

↑ filas

En ajedrez, queremos calcular el número de escaque a partir del entero que identifica la fila y la letra que identifica la columna.

Funciones. Ejemplo de definición

```
/* Pre:  1 ≤ fila ≤ 8 y
 *       'a' ≤ columna ≤ 'h'.
 * Post: Ha devuelto el número de escaque
 *       (entre 1 y 64) que corresponde a
 *       la fila y columnas establecidas por
 *       los parámetros de la función.
 */
int numEscaque(int fila, char columna) {
    return (fila - 1) * 8 + columna - 'a' + 1;
}
```

Funciones. Ejemplos de invocaciones

```
int numEscaque(int fila, char columna) {
    return (fila - 1) * 8 + (columna - 'a') + 1;
}
```

```
...
int primero = numEscaque(1, 'a');
int ultimo = numEscaque(8, 'h');

int fila = 3;
char columna = 'd';
int escaque = numEscaque(fila, columna);

cout << numEscaque(fila + 1, columna - 1) << endl;
```

Funciones. Ejemplos de invocaciones

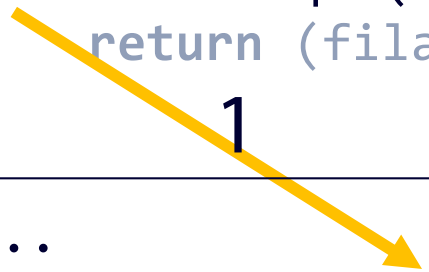
```
int numEscaque(int fila, char columna) {
    return (fila - 1) * 8 + (columna - 'a') + 1;
}
```

1 'a'

```
...
int primero = numEscaque(1, 'a');
int ultimo = numEscaque(8, 'h');

int fila = 3;
char columna = 'd';
int escaque = numEscaque(fila, columna);

cout << numEscaque(fila + 1, columna - 1) << endl;
```

Funciones. Ejemplos de invocaciones

```
int numEscaque(int fila, char columna) {
    return (fila - 1) * 8 + (columna - 'a') + 1;
}
```




```
...
int primero = numEscaque(1, 'a');
int ultimo = numEscaque(8, 'h');

int fila = 3;
char columna = 'd';
int escaque = numEscaque(fila, columna);

cout << numEscaque(fila + 1, columna - 1) << endl;
```

Funciones. Ejemplos de invocaciones

```
int numEscaque(int fila, char columna) {
    return (fila - 1) * 8 + (columna - 'a') + 1;
}
```

8 'h'

```
...
int primero = numEscaque(1, 'a');
int ultimo = numEscaque(8, 'h');

int fila = 3;
char columna = 'd';
int escaque = numEscaque(fila, columna);

cout << numEscaque(fila + 1, columna - 1) << endl;
```

Funciones. Ejemplos de invocaciones

```
int numEscaque(int fila, char columna) {
    return (fila - 1) * 8 + (columna - 'a') + 1;
}
```

64

```
...
int primero = numEscaque(1, 'a');
int ultimo = numEscaque(8, 'h');

int fila = 3;
char columna = 'd';
int escaque = numEscaque(fila, columna);

cout << numEscaque(fila + 1, columna - 1) << endl;
```

Funciones. Ejemplos de invocaciones

```
int numEscaque(int fila, char columna) {
    return (fila - 1) * 8 + (columna - 'a') + 1;
}
```

3 ‘d’

```
...
int primero = numEscaque(1, 'a');
int ultimo = numEscaque(8, 'h');

int fila = 3;
char columna = 'd';
int escaque = numEscaque(fila, columna);

cout << numEscaque(fila + 1, columna - 1) << endl;
```

Funciones. Ejemplos de invocaciones

```
int numEscaque(int fila, char columna) {
    return (fila - 1) * 8 + (columna - 'a') + 1;
}
```

20

```
...
int primero = numEscaque(1, 'a');
int ultimo = numEscaque(8, 'h');

int fila = 3;
char columna = 'd';
int escaque = numEscaque(fila, columna);

cout << numEscaque(fila + 1, columna - 1) << endl;
```

Funciones. Ejemplos de invocaciones

```
int numEscaque(int fila, char columna) {
    return (fila - 1) * 8 + (columna - 'a') + 1;
}
```

20 3 'd'

```
...
int primero = numEscaque(1, 'a');
int ultimo = numEscaque(8, 'h');

int fil = 3;
char col = 'd';
int escaque = numEscaque(fil, col);

cout << numEscaque(fil + 1, col - 1) << endl;
```

Funciones. Ejemplos de invocaciones

```
int numEscaque(int fila, char columna) {
    return (fila - 1) * 8 + (columna - 'a') + 1;
}
```

4 'c'

```
...
int primero = numEscaque(1, 'a');
int ultimo = numEscaque(8, 'h');

int fila = 3;
char columna = 'd';
int escaque = numEscaque(fila, columna);

cout << numEscaque(fila + 1, columna - 1) << endl;
```

Funciones. Ejemplos de invocaciones

```
int numEscaque(int fila, char columna) {
    return (fila - 1) * 8 + (columna - 'a') + 1;
}
```

27

```
...
int primero = numEscaque(1, 'a');
int ultimo = numEscaque(8, 'h');

int fila = 3;
char columna = 'd';
int escaque = numEscaque(fila, columna);

cout << numEscaque(fila + 1, columna - 1) << endl;
```

Índice

- Funciones
- **Estructuración de programas**
  - **Metodología de programación descendente**
- Ámbito y vida
- Comunicación entre funciones

```
¿Qué tabla desea escribir? (0 para acabar): 7

LA TABLA DEL 7

7 x  0 =   0
7 x  1 =   7
7 x  2 =  14
7 x  3 =  21
7 x  4 =  28
7 x  5 =  35
7 x  6 =  42
7 x  7 =  49
7 x  8 =  56
7 x  9 =  63
7 x 10 =  70

¿Qué tabla desea escribir? (0 para acabar): 6

LA TABLA DEL 6

6 x  0 =   0
6 x  1 =   6
6 x  2 =  12
6 x  3 =  18
6 x  4 =  24
6 x  5 =  30
6 x  6 =  36
6 x  7 =  42
6 x  8 =  48
6 x  9 =  54
6 x 10 =  60

¿Qué tabla desea escribir? (0 para acabar): 0
```

```
¿Qué tabla desea escribir? (0 para acabar): 7
LA TABLA DEL 7

7 x  0 =   0
7 x  1 =   7
7 x  2 =  14
7 x  3 =  21
7 x  4 =  28
7 x  5 =  35
7 x  6 =  42
7 x  7 =  49
7 x  8 =  56
7 x  9 =  63
7 x 10 =  70

¿Qué tabla desea escribir? (0 para acabar): 6
LA TABLA DEL 6

6 x  0 =   0
6 x  1 =   6
6 x  2 =  12
6 x  3 =  18
6 x  4 =  24
6 x  5 =  30
6 x  6 =  36
6 x  7 =  42
6 x  8 =  48
6 x  9 =  54
6 x 10 =  60

¿Qué tabla desea escribir? (0 para acabar): 0
```

Una (mala) solución

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

/*
 * Programa que pregunta reiteradamente al usuario qué tabla de multiplicar desea escribir y la escribe
 * a continuación, salvo cuando el usuario responde con un 0, en cuyo caso el programa termina.
 */
int main() {
    cout << "¿Qué tabla desea escribir (0 para acabar)?: ";
    int multiplicando;
    cin  >>  multiplicando;

    while (multiplicando != 0) {
        cout << endl;
        cout << "LA TABLA DEL " << multiplicando << endl;
        for (int i = 0; i <= 10; ++i) {
            cout << setw(3) << multiplicando << " x " << setw(2) << i << " = “
                 << setw(3) << multiplicando * i << endl;
        }
        cout << endl << "¿Qué tabla desea escribir (0 para acabar)?: ";
        cin >> multiplicando;
     }
     return 0;
}
```

Función `main`

```
/*
 * Programa que pregunta reiteradamente al usuario qué
 * tabla de multiplicar desea escribir y la escribe
 * a continuación, salvo cuando el usuario responde con un
 * 0, en cuyo caso el programa termina.
*/
int main() {
   …
}
```

Función `main`

```
int main() {
    // Plantea la primera pregunta al usuario
    cout << "¿Qué tabla desea escribir (0 para acabar)?: ";
    // Asigna a «multiplicando» el primer entero escrito por el usuario
    int multiplicando;
    cin >> multiplicando;

    // Itera hasta que el usuario responda con un valor nulo
    while (multiplicando != 0) {
        // Escribe la tabla de multiplicar de «multiplicando»
        presentarTabla(multiplicando);
        // Plantea una nueva pregunta al usuario
        cout << "¿Qué tabla desea escribir (0 para acabar)?: ";
        // Asigna a «multiplicando» el nuevo valor escrito por el usuario
        cin >> multiplicando;
    }
    return 0;
}
```

```
¿Qué tabla desea escribir? (0 para acabar): 7
LA TABLA DEL 7

7 x  0 =   0
7 x  1 =   7
7 x  2 =  14
7 x  3 =  21
7 x  4 =  28
7 x  5 =  35
7 x  6 =  42
7 x  7 =  49
7 x  8 =  56
7 x  9 =  63
7 x 10 =  70

¿Qué tabla desea escribir? (0 para acabar): 6
LA TABLA DEL 6

6 x  0 =   0
6 x  1 =   6
6 x  2 =  12
6 x  3 =  18
6 x  4 =  24
6 x  5 =  30
6 x  6 =  36
6 x  7 =  42
6 x  8 =  48
6 x  9 =  54
6 x 10 =  60

¿Qué tabla desea escribir? (0 para acabar): 0
```

Función presentarTabla

```
LA TABLA DEL 7

 7 x  0 =   0
 7 x  1 =   7
 7 x  2 =  14
 7 x  3 =  21
 7 x  4 =  28
 7 x  5 =  35
 7 x  6 =  42
 7 x  7 =  49
 7 x  8 =  56
 7 x  9 =  63
 7 x 10 =  70
```

Función presentarTabla

```
/*
 *  Pre:  ---
 *  Post: Ha presentado en la pantalla la tabla de multiplicar del «n»:
 *
 *        LA TABLA DEL «n»
 *        «n» x  0 =  0
 *        «n» x  1 = «n»
 *        «n» x  2 = ...
 *             ...
 *        «n» x  9 = ...
 *        «n» x 10 = ...
 */
void presentarTabla(int n) {
   …
}
```

Función presentarTabla

```
void presentarTabla(int n) {
   // Escribe la cabecera de la tabla de multiplicar del «n»
   cout << endl;
   cout << "LA TABLA DEL " << n << endl;

   // Escribe las 11 líneas de la tabla de multiplicar del «n»
   for (int i = 0; i <= 10; i++) {
      cout << setw(3) << n
           << " x "
           << setw(2) << i
           << " = "
           << setw(3) << n * i
           << endl;
   }
}
```

Programa completo

```
/*
 * Autores: Miguel Ángel Latre y Javier Martínez
 * Última revisión: 20 de marzo de 2014
 * Resumen: Programa interactivo que presenta por pantalla las tablas de
 *          multiplicar seleccionadas por el usuario
 */
#include <iostream>
#include <iomanip>
using namespace std;

/* Pre: … / Post: … */
void presentarTabla(int n) {
    …
}

/* Pre: … / Post: … */
int main() {
    …
}
```

Índice

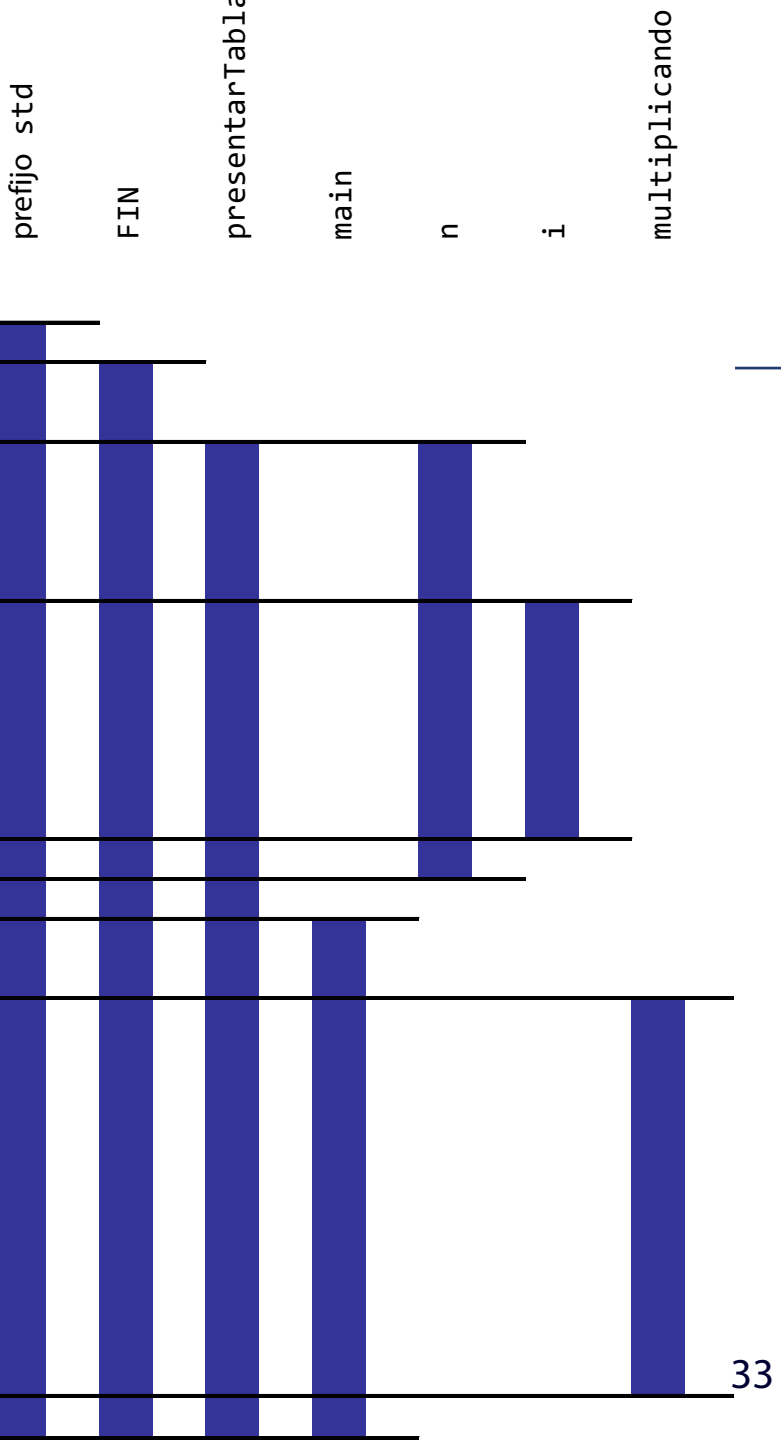
- Funciones
- Estructuración de programas
- **Ámbito y vida**
- Comunicación entre funciones

Ámbito y vida

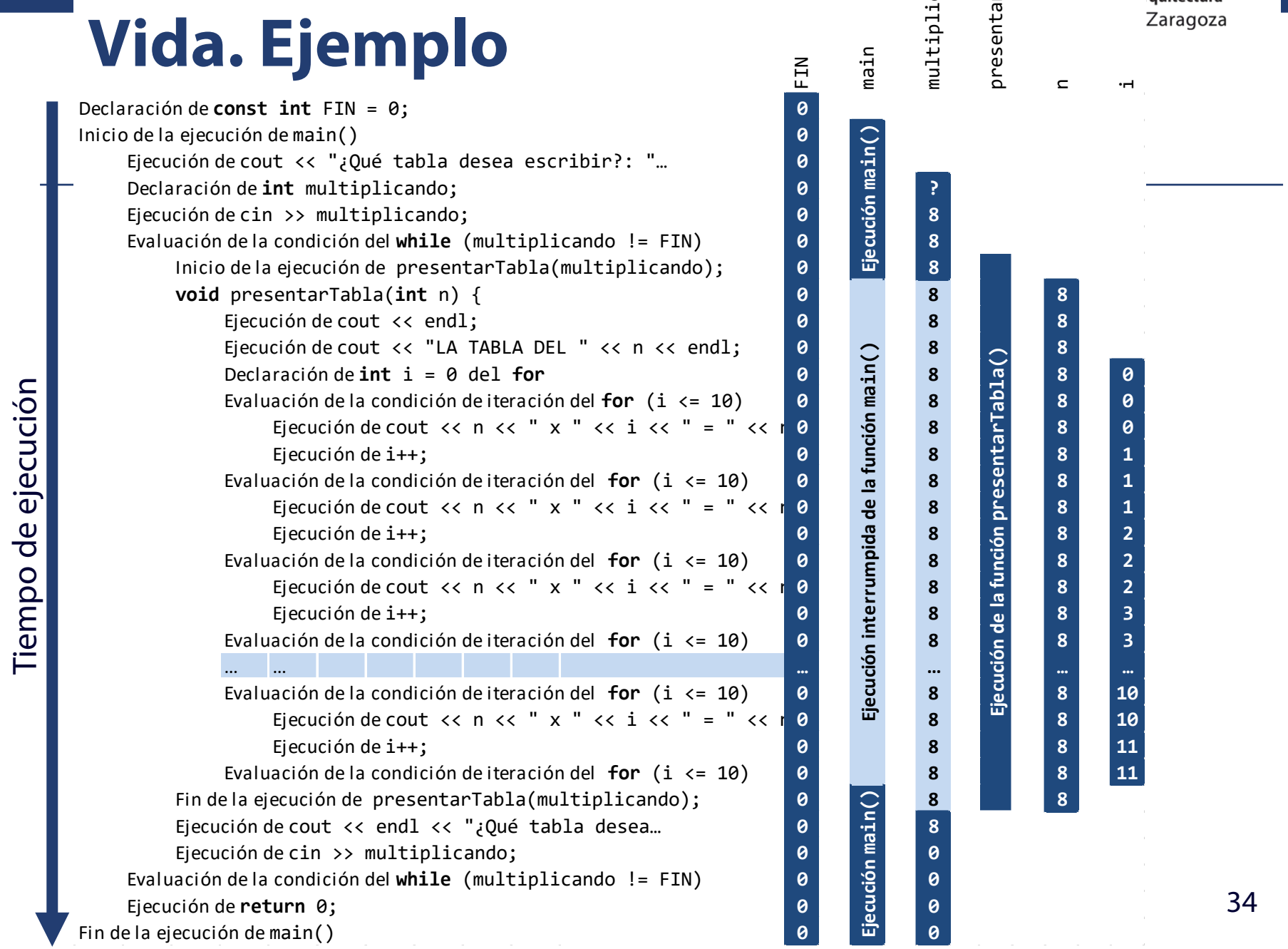
- Elemento nombrado con un identificador: función, constante, variable, parámetro, …
- **Ámbito** o **visibilidad** (*scope*): zona del código en la que un elemento es accesible (se puede hacer uso de él).
  - **Ámbito local** de los elementos definidos dentro de un bloque o función:
    - Desde el punto en que se definen hasta el final del bloque o función.
  - **Ámbito global** de los elementos definidos en el fichero fuera de las funciones:
    - Desde el punto en que se han definido hasta el final del fichero.
- **Duración** o **vida** (*lifetime*) de un elemento
  - Tiempo en el que el elemento está disponible durante la ejecución del programa.

Ámbito. Ejemplo

```
#include <iostream>
#include <iomanip>
using namespace std;
const int FIN = 0;

void presentarTabla(int n) {
     cout << endl;
     cout << "LA TABLA DEL " << n << endl;

     for (int i = 0; i <= 10; i++) {
          cout << setw(3) << n
               << " x " << setw(2) << i
               << " = " << setw(3) << n * i
               << endl;
     }
}

int main() {
     cout << "¿Qué tabla desea escribir?: "…
     int multiplicando;
     cin  >>  multiplicando;
     while (multiplicando != FIN) {
          presentarTabla(multiplicando);
          cout << endl << "¿Qué tabla desea…
          cin >> multiplicando;
     }

     return 0;
}
```



Vida. Ejemplo

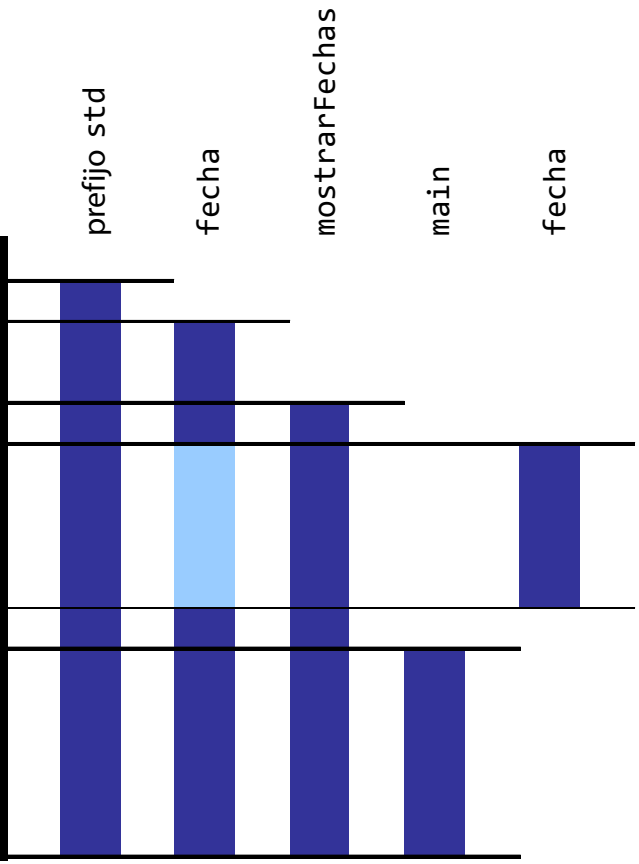


Ámbito. Ejemplo de enmascaramiento

```
#include <iostream>
using namespace std;
int fecha = 1492;

void mostrarFechas() {
   int fecha = 1969;
   cout << fecha << endl;
}

int main() {
    mostrarFechas();
    return 0;
}
```

Ámbito. Ejemplo de enmascaramiento

```
#include <iostream>
using namespace std;
int fecha = 1492;
void mostrarFechas() {
   int fecha = 1969;

   cout << fecha << endl;

}

int main() {
    mostrarFechas();
    return 0;
}
```



Espacios de nombres

```
namespace guignote {
   enum Palo { OROS, COPAS, ESPADAS, BASTOS };
   enum Valor { DOS, CUATRO, CINCO, SEIS, SITE,
                CABALLO, SOTA, REY, TRES, AS };

   struct Carta {
      Palo palo;
      Valor valor;
   };

   bool puedeMatar(Carta carta, Carta mesa, Palo triunfo);
   int contarPuntos(Carta bazas[]);
}
```

Índice

- Funciones
- Estructuración de programas
- Ámbito y vida
- **Comunicación entre funciones**

Comunicación entre funciones

- Parámetros por valor — Ya vistos
- Parámetros por referencia — En este tema
- Valor devuelto — Ya visto
- Variables globales

Comunicación entre funciones

- Parámetros por valor
- Parámetros por referencia
- Valor devuelto
- Variables globales
  - **¡PROHIBIDAS EN ESTE CURSO!**

Diseño de un programa

```
Escriba los extremos de un intervalo entero
[a, b] siendo a <= b: 100 150
Los enteros del intervalo [100, 150] suman 6375.
```

Diseño de un programa

- Lo vamos a resolver utilizando tres funciones:
  - `main`
    - Pide al usuario que defina los extremos de un intervalo entero y presenta en la pantalla el valor de la suma de todos los enteros de dicho intervalo.
  - `sumarDatos`
    - Calcula la suma de los datos de un intervalo de enteros.
  - `mostrarResultado`
    - Escribe en la pantalla de la suma de los datos de un intervalo de enteros a través de un mensaje de este tipo: "Los enteros del intervalo [*100*, *150*] suman *6375*."

Comunicación de datos mediante variables globales

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

int inicial, final;          // datos globales: extremos del un intervalo

/* Pre: … Post: …  */
int sumarDatos() { … }


/* Pre: … Post: …  */
void mostrarResultado() { … }


/* Pre: … Post: …  */
int main() { … }
```

Comunicación de datos mediante variables globales

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

int inicial, final;          // datos globales: extremos del un intervalo
...
/*
 * Pre: --
 * Post: Ha pedido al usuario que defina los extremos de un intervalo entero y
 *       ordena presentar por pantalla el valor de la suma de todos los
 *       enteros de dicho intervalo.
 */
int main() {
    cout << "Escriba los extremos de un intervalo entero [a, b] siendo a <= b: ";
    cin >> inicial >> final;
    mostrarResultado();
    return 0;
}
```

Comunicación de datos mediante variables globales

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

int inicial, final;          // datos globales: extremos del un intervalo
...

/*
 * Pre:  inicial <= final
 * Post: Ha informado por la pantalla de la suma de los datos del
 *       intervalo entero [inicial, final] del siguiente modo, por
 *       ejemplo:
 *           Los enteros del intervalo [100, 150] suman 6375.
 */
void mostrarResultado() {
    cout << "Los enteros del intervalo [" << inicial << ", " << final
         << "] suman " << sumarDatos() << '.' << endl;
}
...
```

Comunicación de datos mediante variables globales

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

int inicial, final;          // datos globales: extremos del un intervalo

/*
 * Pre:  inicial <= final
 * Post: Ha devuelto la suma de los datos del intervalo entero
 *       [inicial, final].
 */
int sumarDatos() {
    return (inicial + final) * (final - inicial + 1) / 2;
}
...
```

Problemas del uso de variables globales

- Diseño dependiente del nombre de las variables globales
  - Reducción de la capacidad de reutilización del código
- Efectos laterales debidos a la posibilidad de modificación de sus valores desde cualquier parte del código
- Reducción de la legibilidad de las funciones

Comunicación mediante parámetros por valor

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

/* Pre: …
 * Post: …  */
int sumarDatos(int inicial, int final) { … }

/* Pre: …
 * Post: …  */
void mostrarResultado(int principio, int fin) { … }

/* Pre: …
 * Post: …  */
int main() { … }
```

Comunicación mediante parámetros por valor

```
/*
 * Programa que pide al usuario que defina los extremos de un intervalo
 * entero y ordena presentar por pantalla el valor de la suma de
 * todos los enteros de dicho intervalo.
 */
int main() {
    cout << "Escriba los extremos de un intervalo entero [a, b] "
         << "siendo a <= b: ";
    int minimo, maximo;
    cin >> minimo >> maximo;
    mostrarResultado(minimo, maximo);
    return 0;
}
```

Comunicación mediante parámetros por valor

```
/*
 * Pre:  principio <= fin
 * Post: Ha informado por la pantalla de la suma de los datos
 *       del intervalo entero [principio, fin] del siguiente
 *       modo: (ejemplo):
 *           Los enteros del intervalo [100,150] suman 6375.
 */
void mostrarResultado (int principio, int fin) {
   cout << "Los enteros del intervalo [" << principio << ","
        << fin << "] suman " << sumaDatos(principio, fin) << '.'
        << endl;
}
```

Comunicación mediante parámetros por valor

```
/*
 * Pre:  inicial <= final
 * Post: Devuelve la suma de los datos comprendidos en el
 *       intervalo entero [inicial, final]
 */
int sumaDatos(int inicial, int final) {
    return (inicial + final) * (final - inicial + 1) / 2;
}
```

Otro problema distinto

```
int a, b;
...
// Si ahora: a = X y b = Y …
intercambiar(a, b);
// … entonces ahora: a = Y y b = X
```

Comunicación por valor. Solución errónea

```
/*
 * Pre:  uno = X0 y otro = Y
 * Post: uno = Y0 y otro = X0
 */
void intercambiar(int uno, int otro) {
  int aux = uno;
  uno = otro;
  otro = aux;
}
```

Comunicación mediante parámetros por referencia

```
/*
 * Pre:  uno = X0 y otro = Y0
 * Post: uno = Y0 y otro = X0
 */
void intercambiar(int& uno, int& otro) {
  int aux = uno;
  uno = otro;
  otro = aux;
}
```

Comunicación mediante parámetros por referencia

```
/*
 * Pre:  a = X y b = Y
 * Post: «a» almacena el menor de los
 *       valores {X, Y} y «b» almacena el
 *       mayor de los valores {X, Y}.
 */
void ordenar(int& a, int& b) {
  if (a > b) {
    intercambiar(a, b);
  }
}
```

Comunicación mediante parámetros por referencia

```
/*
 * Pre:  a = X, b = Y y c = Z
 * Post: «a» almacena el menor de los valores
 *       {X, Y, Z}, «c» almacena el mayor de
 *       los valores {X, Y, Z} y «b» almacena el
 *       valor intermedio de {X, Y, Z}.
 */
void ordenar(int& a, int& b, int& c) {
  ordenar(a, b);
  ordenar(b, c);
  ordenar(a, b);
}
```

Comunicación mediante parámetros por referencia

```
/*
 * Pre:  ---
 * Post: Ha asignado a los parámetros «nacimiento», «estatura» y «peso» los
 *       valores determinados por el usuario como respuesta a tres preguntas
 *       que le son formuladas acerca de su año de nacimiento, su
 *       estatura y su peso.
*/
void preguntarDatos(int& nacimiento, double& estatura, double& peso) {
    cout  << "Escriba año de nacimiento: ";          // 1.ª pregunta
    cin >> nacimiento;                               // lee la respuesta
    cout  << "Su estatura: ";                        // 2.ª pregunta
    cin >> estatura;                                 // lee la respuesta
    cout  << "Su peso: ";                            // 3.ª pregunta
    cin >> peso;                                     // lee la respuesta
}
```

Índice

- Funciones
- Estructuración de programas
- Ámbito y vida
- Comunicación entre funciones