Programación 1

Tema 7

Desarrollo modular y descendente de programas

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

Índice

- Programas dirigidos por menú
- Diseño modular
- Módulos de biblioteca en C++

Programa dirigido por menú

```
MENÚ DE OPERACIONES
===================
0 - Finalizar
1 - Calcular el número de cifras de un entero
2 - Sumar las cifras de un entero
3 - Extraer una cifra de un entero
4 - Calcular la imagen especular de un entero
5 - Comprobar si un entero es primo

Seleccione una operación [0-5]: 4
Escriba un número entero: 8802361
El número imagen especular del 8802361 es el 1632088

...
```

Programa dirigido por menú

```
...
MENÚ DE OPERACIONES
====================
0 - Finalizar
1 - Calcular el número de cifras de un entero
2 - Sumar las cifras de un entero
3 - Extraer una cifra de un entero
4 - Calcular la imagen especular de un entero
5 - Comprobar si un entero es primo

Seleccione una operación [0-5]: 5
Escriba un número entero: 103
El número 103 es primo
...
```

Programa dirigido por menú

```
...
MENÚ DE OPERACIONES
====================
0 - Finalizar
1 - Calcular el número de cifras de un entero
2 - Sumar las cifras de un entero
3 - Extraer una cifra de un entero
4 - Calcular la imagen especular de un entero
5 - Comprobar si un entero es primo

Seleccione una operación [0-5]: 7
Opción desconocida
...
```

Programa dirigido por menú

```
...
MENÚ DE OPERACIONES
====================
0 - Finalizar
1 - Calcular el número de cifras de un entero
2 - Sumar las cifras de un entero
3 - Extraer una cifra de un entero
4 - Calcular la imagen especular de un entero
5 - Comprobar si un entero es primo

Seleccione una operación [0-5]: 0
```

Estructura modular

- **Programas grandes**
  - Descomposición en **módulos**
    - Permiten desarrollo independiente (no necesariamente por un único programador)
- **Módulo de programa**
  - Contiene el código de la función principal del programa
- **Módulos de biblioteca**
  - Módulos adicionales en los que se puede dividir un programa y con los que puede contar

Estructura modular en C++

- **Módulo principal** obligatorio
  - Se define en él, al menos, la función `main`
  - Se almacena en un fichero con sufijo `.cc` o `.cpp`
- **Módulos de biblioteca**
  - Definen recursos puestos a disposición de otros módulos
    - Tipos de datos
    - Datos constantes [y variables]
    - Funciones

Estructura modular en C++

- **Módulos de biblioteca**
  - Constan de dos ficheros:
    - **Interfaz del módulo**
      - Declaraciones y especificaciones de los recursos visibles fuera del módulo
      - Se almacena en un **fichero de cabecera**, un fichero con sufijo `.h`
    - **Implementación del módulo**
      - Código de las funciones declaradas en la interfaz
      - Elementos auxiliares
      - Se almacena en un fichero con sufijo `.cc` o `.cpp`

Programa del ejemplo

- Diseño con una estructura modular aplicando una metodología descendente:
  - Módulo principal
    - Fichero `calculadora-main.cpp`
    - Gestiona la interacción con el usuario con un comportamiento iterativo:
      - Plantea el menú de opciones (operaciones disponibles).
      - Lee la opción seleccionada por el usuario.
      - Ejecuta la orden correspondiente a la opción elegida por el usuario.
  - Módulo de biblioteca `calculos`
    - Define siete funciones que realizan cálculos y análisis de propiedades de enteros.

Programa del ejemplo

- Módulo de biblioteca `calculos`
  - Define siete funciones que realizan cálculos y análisis de propiedades de enteros:
    - `int numCifras(int n)`
    - `int sumaCifras(int n)`
    - `int cifra(int n, int i)`
    - `int imagen(int n)`
    - `int factorial(int n)`
    - `bool esPrimo(int n)`
    - `int mcd(int a, int b)`
  - Compuesto por dos ficheros
    - Interfaz del módulo: fichero de cabecera `calculos.h`
    - Implementación del módulo: fichero `calculos.cpp`

Diseño descendente. Módulo principal. Primer nivel

```
/* Pre:  --- // Post: Ha planteado al usuario de forma … */
int main() {
    int operacion;
    pedirOrden(operacion);

    // Itera hasta que el valor de «operacion» sea igual a 0.
    while (operacion != 0) {
        ejecutarOrden(operacion);
        pedirOrden(operacion);
    }

    return 0;
}
```

Diseño descendente. Módulo principal. Segundo nivel

```
/*
 *  Pre:  ---
 *  Post: Ha presentado en la pantalla el menú de opciones disponibles,
 *        ha solicitado al usuario que escriba el código de una de ellas
 *        y ha asignado a «operacion» la nueva respuesta del usuario.
 */
void pedirOrden(int& operacion) {
    presentarMenu();
    cout << "Seleccione una operacion [0-" << NUM_OPERACIONES << "]: ";
    cin >> operacion;
}
```

Diseño descendente. Módulo principal. Tercer nivel

```
/*
 *  Pre:  ---
 *  Post: Presenta el menú de opciones disponibles
 */
void presentarMenu() {
     cout << endl;
     cout << "MENU DE OPERACIONES" << endl;
     cout << "===================" << endl;
     cout << "0 - Finalizar" << endl;
     cout << "1 - Calcular el numero de cifras de un entero" << endl;
     cout << "2 - Sumar las cifras de un entero" << endl;
     cout << "3 - Extraer una cifra de un entero" << endl;
     cout << "4 - Calcular la imagen especular de un entero" << endl;
     cout << "5 - Comprobar si un entero es primo" << endl << endl;
}
```

Diseño descendente. Módulo principal. Segundo nivel

```
/*
 * Pre:  ---
 * Post: Ejecuta las acciones asociadas a la orden cuyo código es
 *       «operacion».
 */
void ejecutarOrden(int operacion) {
    if (operacion >= 1 && operacion <= 5) {
        // Se va a ejecutar una operación válida.
        // En primer lugar se pide al usuario que defina un número entero.
        cout << "Escriba un número entero: ";
        int numero;
        cin >> numero;

        if (operacion == 1) {
            ejecutarNumCifras(numero);
        }
        else if (operacion == 2) {...}
        ...
    }
    else {
        // El código de operación no es válido
        cout << "Opción desconocida" << endl;
    }
}
```

Diseño descendente. Módulo principal. Tercer nivel

```
/*
 * Pre:  ---
 * Post: Ha ejecutado la 1ª orden,
 *       informando del número de cifras
 *       de «numero».
 */
void ejecutarNumCifras(int numero) {
    cout << "El número " << numero
         << " tiene " << numCifras(numero)
         << " cifras." << endl;
}
```

Diseño descendente. Estructura del módulo principal

```
#include <iostream>
#include "calculos.h"
using namespace std;

void presentarMenu() {...}

void pedirOrden(int& operacion) {...}

void ejecutarNumCifras(int numero) {...}
...
void ejecutarOrden(int operacion) {...}

int main() {...}
```

Diseño descendente.
4.º nivel. Módulo `calculos`. Interfaz

```
/*
 * Pre:  ---
 * Post: Ha devuelto el número de cifras de «n» cuando este
 *       número se escribe en base 10.
 */
int numCifras(int n);

/*
 * Pre:  ---
 * Post: Ha devuelto la suma de las cifras de «n» cuando «n» se
 *       escribe en base 10.
 */
int sumaCifras(int n);

...
```

Diseño descendente. 4.º nivel. Módulo `calculos`. Implementación

```c
#include "calculos.h"

/**
  * Pre:  ---
  * Post: Ha devuelto el número de cifras de «n» cuando este número se
  *       escribe en base 10.
  */
int numCifras(int n) {
   int cuenta = 1; n = n / 10;
   while (n != 0) {
      cuenta++; n = n / 10;
   }
   return cuenta;
}

/**
 * Pre:  ---
 * Post: Ha devuelto la suma de las cifras de «n» cuando «n» se escribe
 *       en base 10.
 */
int sumaCifras(int n) {
    ...
}
```

Diseño modular del programa

Módulo principal

calculadora-main.cpp

```
#include <iostream>
#include "calculos.h"

void presentarMenu() {…}
void ejecutarOrden(int operacion) {…}
int main() {…}
```

Módulo `calculos`

calculos.h

```
int numCifras(int n);
int sumaCifras(int n);
int cifra(int n, int i);
int imagen(int n);
int factorial(int n);
bool esPrimo(int n);
int mcd(int a, int b);
```

calculos.cpp

```
#include "calculos.h"

int numCifras(int n) {…}
int sumaCifras(int n) {…}
int cifra(int n, int i) {…}
int imagen(int n) {…}
int factorial(int n) {…}
bool esPrimo(int n) {…}
int mcd(int a, int b) {…}
```

Compilación modular

calculos.h

```
int numCifras(int n);
int sumaCifras(int n);
int cifra(int n, int i);
int imagen(int n);
int factorial(int n);
bool esPrimo(int n);
int mcd(int a, int b);
```

calculadora-main.cpp

```
#include <iostream>
#include "calculos.h"

void presentarMenu() {…}
void ejecutarOrden(int operacion) {…}
int main() {…}
```

calculos.cpp

```
#include "calculos.h“

int numCifras(int n) {…}
int sumaCifras(int n) {…}
int cifra(int n, int i) {…}
int imagen(int n) {…}
int factorial(int n) {…}
bool esPrimo(int n) {…}
int mcd(int a, int b) {…}
```

Compilación modular

1.ª compilación (`calculos.cpp`)

calculos.h

```
int numCifras(int n);
int sumaCifras(int n);
int cifra(int n, int i);
int imagen(int n);
int factorial(int n);
bool esPrimo(int n);
int mcd(int a, int b);
```

calculadora-main.cpp

```
#include <iostream>
#include "calculos.h"

void presentarMenu() {…}
void ejecutarOrden(int operacion) {…}
int main() {…}
```

calculos.cpp

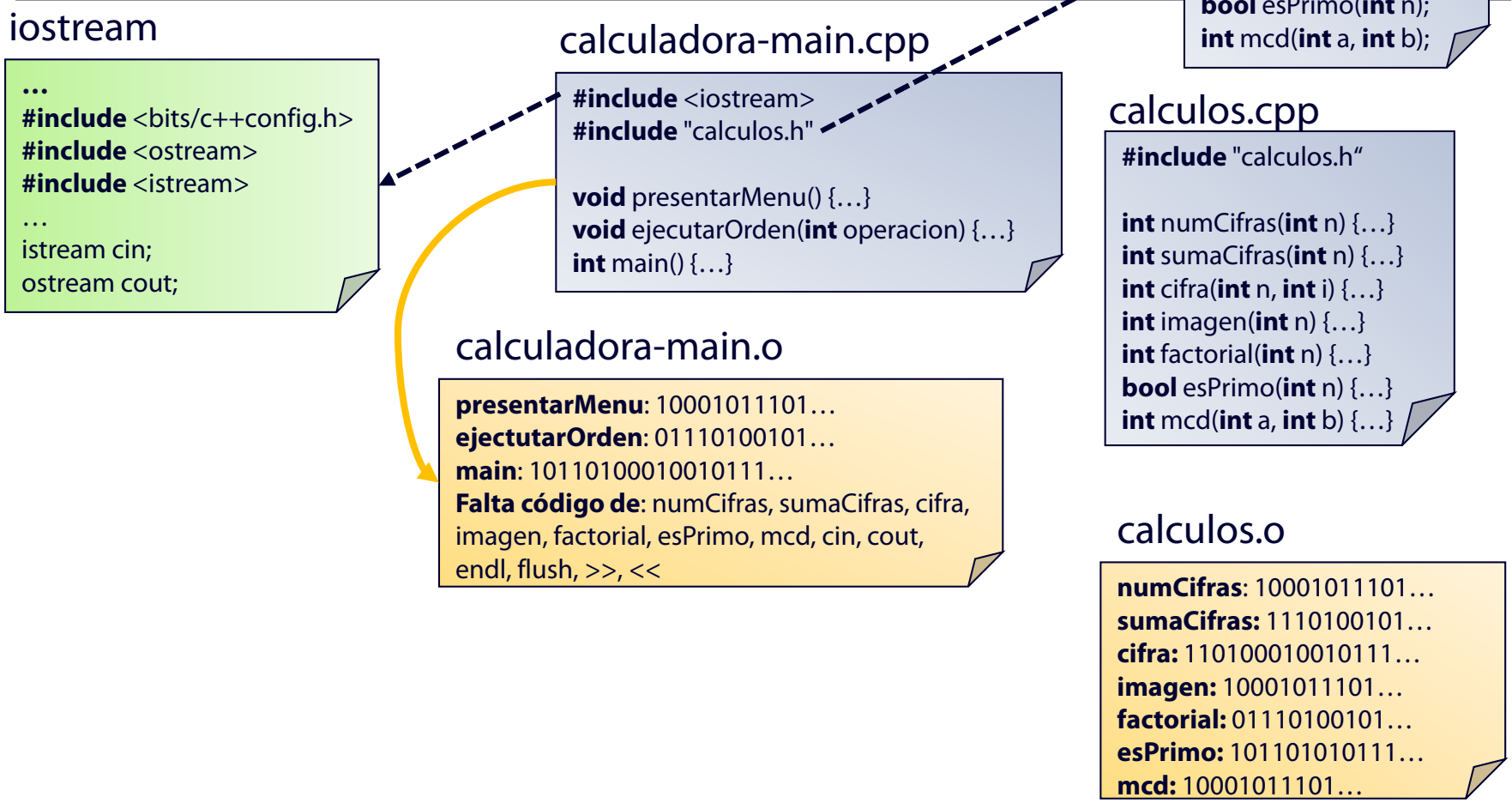
```
#include "calculos.h"

int numCifras(int n) {…}
int sumaCifras(int n) {…}
int cifra(int n, int i) {…}
int imagen(int n) {…}
int factorial(int n) {…}
bool esPrimo(int n) {…}
int mcd(int a, int b) {…}
```

calculos.o

```
numCifras: 10001011101…
sumaCifras: 1110100101…
cifra: 110100010010111…
imagen: 10001011101…
factorial: 01110100101…
esPrimo: 101101010111…
mcd: 10001011101…
```

Compilación modular

2.ª compilación (`calculadora-main.cpp`)



Compilación modular

3.ª compilación (enlazado)

calculos.h

```
int numCifras(int n);
int sumaCifras(int n);
int cifra(int n, int i);
int imagen(int n);
int factorial(int n);
bool esPrimo(int n);
int mcd(int a, int b);
```

iostream

```
...
#include <bits/c++config.h>
#include <ostream>
#include <istream>
…
istream cin;
ostream cout;
```

calculadora-main.cpp

```
#include <iostream>
#include "calculos.h"

void presentarMenu() {…}
void ejecutarOrden(int operacion) {…}
int main() {…}
```

calculos.cpp

```
#include "calculos.h“

int numCifras(int n) {…}
int sumaCifras(int n) {…}
int cifra(int n, int i) {…}
int imagen(int n) {…}
int factorial(int n) {…}
bool esPrimo(int n) {…}
int mcd(int a, int b) {…}
```

libstdc++.a

```
100010111011110100
101110100010010111
1000101110101110…
```

calculadora-main.o

presentarMenu: 10001011101…
ejectutarOrden: 01110100101…
main: 10110100010010111…
Falta código de: numCifras, sumaCifras, cifra, imagen, factorial, esPrimo, mcd, cin, cout, endl, flush, >>, <<

calculos.o

numCifras: 10001011101…
sumaCifras: 1110100101…
cifra: 110100010010111…
imagen: 10001011101…
factorial: 01110100101…
esPrimo: 101101010111…
mcd: 10001011101…

calculadoraEnteros.exe

```
MZ10001011101111010010111010001001011110001011101011101001010…
```