Estructura de un programa en C

Departamento de Automática
Universidad de Alcalá

/gso>

Índice

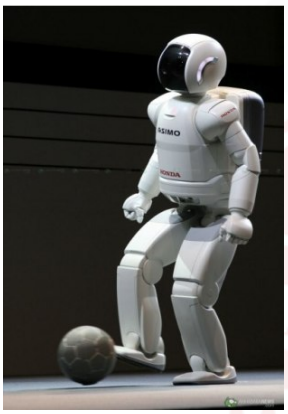
Introducción a la programación

¿Por qué programar?

- Nuestra civilización funciona sobre software
  - Casi todas las actividades de la ingeniería implican software: Diseño, control, monitorización, simulación, optimización, ...
- La mayor parte del software no es visible
  - Los programas NO suelen correr sobre un PC con pantalla, teclado y una caja bajo la mesa
  - Una minoría de los programas son parecidos a la ofimática
  - Los programas corren sobre microprocesadores
  - Los micros se integran en casi todo: Sistemas empotrados
- Programar es necesario porque:
  - Casi todo sistema digital está programado
  - El *software aporta inteligencia a los dispositivos*
  - Razones pedagógicas: Estructura la mente y potencia pensamiento lógico
  - ... y además, programar es divertido

Introducción a la programación

¿Por qué programar? Áreas de aplicación: Robótica



- Control
- Visión artificial
- Inteligencia artificial
- Monitorización
- Automatización

Introducción a la programación

¿Por qué programar? Áreas de aplicación: Aeroespacial

- Comunicaciones
- Navegación
- Diseño de ala y motores

- Procesado de señal
- Navegación
- Planificación de la misión

Introducción a la programación

¿Por qué programar? Áreas de aplicación: Informática personal

¡Hasta sirve para hacer procesadores de texto y videojuegos!

Introducción a la programación
¿Por qué usar C?

Características de C

- Es pequeño, eficiente y estable
- Hay mucho código C escrito
- C es la base de muchos otros lenguajes
  - C++, Java, AWK, PHP, ...
- C es muy usado en sistemas empotrados
  - También se usan otros lenguajes: C++, ADA, e incluso Java
- No todo son ventajas
  - Bajo nivel (puede ser ventaja)
  - Poco estructurado: Difícil de aprender

Introducción a la programación
Vocabulario básico de programación

Términos comunes en programación

- *Código fuente*: Lo que escribe el programador
- *Compilar*: Generar un ejecutable a partir del código fuente
- *Ejecutable*: Programa en código máquina que se ejecuta
- *Biblioteca*: Funciones externas que realizan ciertas tareas
- *Palabra*: Según el micro, 16, 32 ó 64 bits
- *Algoritmo*: Secuencia de acciones para solucionar un problema

Términos importantes en Ingeniería Industrial

- *Sistema empotrado*: Ordenador insertado en un objeto
- *Sistema crítico*: Sistema que *nunca* debe fallar
- *Sistema de tiempo real*: Sistema con restricciones de tiempo

Conceptos básicos
Historia de C

- Creado por Brian Kernighan y Dennis Ritchie en los laboratorios AT&T
- Originariamente se creó para codificar UNIX
  - UNIX fue creado por Ritchie junto con Ken Thompson
  - Programar un SO en un lenguaje de alto nivel fue revolucionario
  - UNIX fue portado a distintas máquinas y plataformas
- Originariamente C se describió en la primera edición del K&R (1978)
- El K&R fue usado como estándar de facto de C

Conceptos básicos
Estándares de C

- **1988**: Segunda edición del K&R, actualizado con ANSI C, incorpora biblioteca estándar
- **1989 (C89)**: Nuevas órdenes para el preprocesador y nuevas palabras reservadas: `const`, `volatile`, declaraciones y chequeo de tipos
- **1995 (C89)**: Nuevas cabeceras: `iso646.h`, `wctype.h`, textttwchar.h, ampliación de `printf/scanf`, nuevas funciones y tipos
- **1999 (C99 or ISO/IEC 9899)**: Vectores de tamaño variable, tipos booleanos, mejor soporte para caractéres no ingleses, mejor soporte de coma flotante y comentarioes estilo C++
- **2011 (C11 or ISO/IEC 9899:2011)**: Define `noreturn`, elimina `gets`, estructuras y uniones anónimas, aserciones estáticas

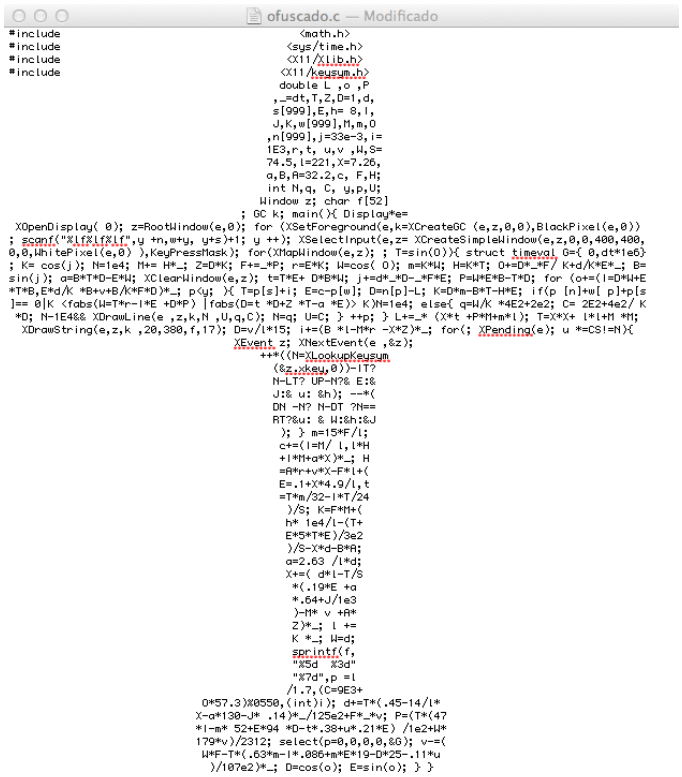
Conceptos básicos
Primer contacto (I)

Hola, mundo

```c
/*
 * Mi programa en C
 */
#include<stdio.h>

int main() {
    printf("Hola, mundo\n");

    return 0; // Fin
}
```

- `include` define funciones externas
  - En este caso `stdio.h`
- El programa empieza en `main()`
- `printf` imprime una cadena
- `return` finaliza la ejecución
- // y /* ... */ son comentarios
- C ignora fin de línea
  - El ; indica fin de instrucción

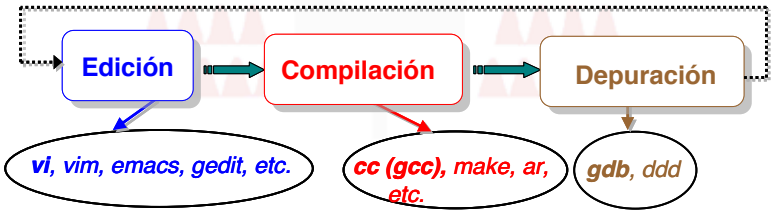
Conceptos básicos
Primer contacto (II)



Conceptos básicos
Fases de desarrollo de un programa

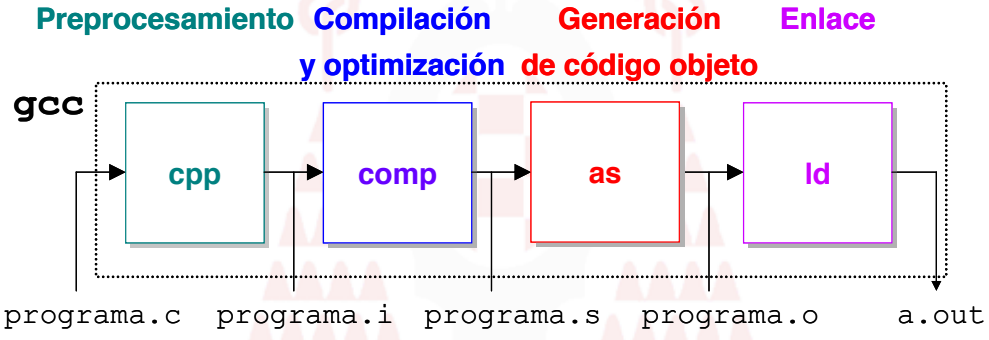
Ciclo de desarrollo

1. Edición
2. Compilación
3. Ejecución
4. Depuración

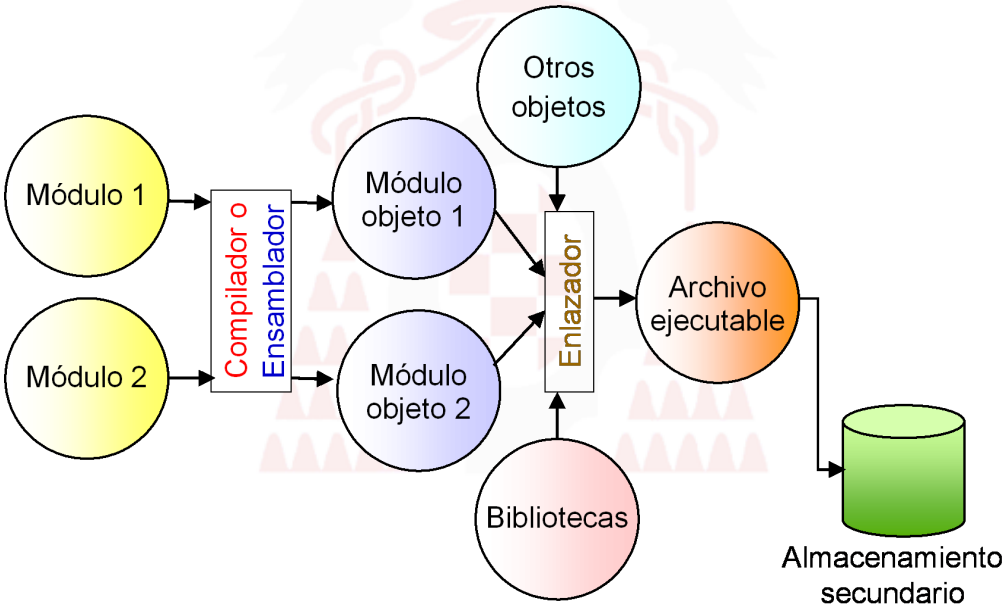



```
$ vi enteros.c → $ gcc enteros.c -o enteros → $ gdb enteros
```


Conceptos básicos
Fases de desarrollo de un programa: Compilación



Conceptos básicos
Visión general



Elementos básicos de C
Variables (I)

- Los programas necesitan almacenar datos
- Cada dato se guarda en una variable
  - Todas las variables se definen al principio
  - La definición indica el tipo de dato

Tipos de datos básicos en C

- Entero: int
- Coma flotante: float
- Carácter: char

Hay otros muchos tipos de datos

Elementos básicos de C
Variables (II)

Cálculo de una media

```
#include<stdio.h>

int main() {
    int var1=3, var2=5;
    float var3=1.5, resultado;

    resultado = (var1 + var2 + var3) / 3;

    printf("La media de %d, %d y %d es %f\n",
        var1, var2, var3, resultado);

    return 0;
}
```


Elementos básicos de C
Constantes (I)

- A veces se necesita usar valores constantes
- Conviene definirlas como tales
  - ¡Mejor no usar variables!

Constantes en C

#define valor contenido

- `valor` es igual a `contenido`
- `valor`, por convenio, se pone en mayúsculas

Elementos básicos de C
Constantes (II)

Cálculo del área de un círculo

```
#include<stdio.h>

#define PI 3.14159265

int main() {
    float radio = 1.5;
    float area;

    area = PI * radio * radio;
    printf("Radio=%f, area=%f", radio, area);

    return 0;
}
```


Elementos básicos de C
Sentencias de control (I)

- Los programas necesitan repetir operaciones
- Un bucle es una porción de código que se repite
  - Todo bucle tiene una condición
  - La condición determina si se sigue repitiendo o no
- ¡Cuidado con los bucles infinitos!

Tipos de bucle

- while: Mientras se cumple una condición
- for: Para una condición inicial, mientras se cumpla, se hace una acción

Elementos básicos de C
Sentencias de control (II)

Ejemplo de bucle (versión 1)

```
#include<stdio.h>

int main() {
    int variable = 0;

    while(variable < 5) {
        printf("Variable = %d\n");
        variable++;
    }
}
```


Elementos básicos de C
Sentencias de control (III)

Ejemplo de bucle (versión 2)

```
#include<stdio.h>

int main() {
    int variable;

    for(variable=0; variable<5; variable++) {
        printf("Variable = %d\n", variable);
    }
}
```

Elementos básicos de C
Sentencias de control (IV)

- A veces interesa ejecutar o no código según una condición
  - Puede ser numérica o lógica
- ¡Cuidado con los bucles infinitos!

Ejecución condicional

- if: Ejecuta código según una condición
- else: Ejecuta código si la condición no se cumple (optativo)

Elementos básicos de C
Sentencias de control (V)

Ejemplo de if (versión 1)

```
int edad = 19;
if (edad > 18) {
    printf("Mayor de edad");
} else {
    printf("Menor de edad");
}
```

Ejemplo de if (versión 2)

```
int num = 20;
if ((num > 10) && (num%2 == 0)) {
    printf("num es mayor de 10 y par");
}
```

Elementos básicos de C
Vectores (I)

- También llamados *arrays*
- Son datos consecutivos del mismo tipo
  - Similar a un vector matemático
    $\vec{v} = (a_0, a_1, a_2, ..., a_n)$
  - Se accede por índice: $v[3]$
- Los elementos se manejan como variables

int v[4]

v[0]	v[1]	v[2]	v[3]
3	5	2	0

Definición de vector

tipo nombre[tamaño];

- Ejemplo: *int vector[10];*
- Ejemplo: *vector[2] = 4;*
- Ejemplo: *printf("$d", vector[2]);*

Elementos básicos de C
Vectores (II)

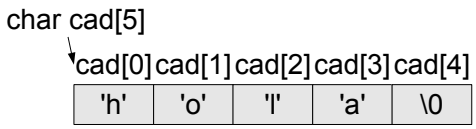
Ejemplo de vector

```
int vector[10];
int i;

for (i=0; i<10; i++) {
  vector[i] = 8*i;
}

printf("Tabla del 8");

for (i=0; i<10; i++) {
  printf("8 x %d = %d\n", i, vector[i]);
}
```

Elementos básicos de C

Vectores (III): Cadenas de caracteres

- Las cadenas de caracteres se manejan como vectores
- Definición:
  char nombre[tamaño];
  - Fin de cadena se marca con '\0'
  - Tamaño = longitud + 1



Ejemplo de cadena (versión 1)

```
char cadena[5];

cadena[0]='h'; cadena[1]='o';
cadena[2]='l'; cadena[3]='a';
cadena[4]='\0';
printf("%s", cadena);
```

Ejemplo de cadena (versión 2)

```
char cadena = "hola";

printf("%s", cadena);
```


Elementos básicos de C

Funciones (I)

- Un programa típico contiene mucho código
  - Incluso millones de líneas
  - Descomponer el problema
  - *Divide y vencerás*
- Las funciones son trozos de código reutilizables
  - Implementan una tarea
  - Función especial: `main()`
- Las funciones pueden devolver un valor

Ejemplo

```
void imprimir() {
  printf("Hola, mundo");
}

int main() {
  imprimir();

  return 0;
}
```

Regla: *Meter en una función el código que se use más de una vez*

Elementos básicos de C
Funciones (II)

Ejemplo con funciones

```
int sumar(int a, int b) {
  int c = a + b;

  return c;
}
int main() {
  int var1 = 3, var2=2, resultado;

  resultado = sumar(a,b);
  printf("Resultado = %d", resultado);

  return 0;
}
```