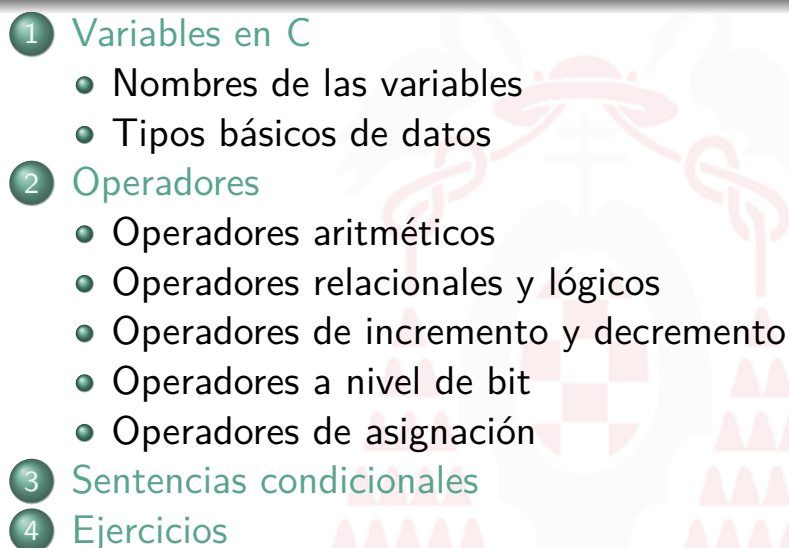


Elementos del lenguaje C

Departamento de Automática
Universidad de Alcalá



Índice

- 
- 1 Variables en C
 - Nombres de las variables
 - Tipos básicos de datos
 - 2 Operadores
 - Operadores aritméticos
 - Operadores relacionales y lógicos
 - Operadores de incremento y decremento
 - Operadores a nivel de bit
 - Operadores de asignación
 - 3 Sentencias condicionales
 - 4 Ejercicios

Variables en C

Nombres de las variables

- Se pueden usar letras y dígitos
- Mayúsculas y minúsculas son diferentes
 - C es *case sensitive*
 - Variables en minúsculas, constantes en mayúsculas
- No se pueden usar palabras reservadas
- Se recomienda poner nombres identificativos

Nombres válidos

resultado, a, i2,
_resultado.
variableTemporal1

Palabras reservadas

if, else, while, for,
case, int, float,
unsigned, ...

Variables en C

Tipos básicos de datos (I)

| Definición | Tipo | Tam. | Desde | Hasta |
|----------------|--------|-------|------------------------|-----------------------|
| char | Entero | 8 | -127 | 127 |
| unsigned char | Entero | 8 | 0 | 255 |
| short | Entero | 16 | -32.768 | 32.767 |
| unsigned short | Entero | 16 | 0 | 65.535 |
| int | Entero | 32 | -2.147.483.648 | 2.147.483.647 |
| unsigned int | Entero | 32 | 0 | 4.294.967.295 |
| long | Entero | 32/64 | -2.147.483.648 | 2.147.483.647 |
| unsigned long | Entero | 32/64 | 0 | 4.294.967.295 |
| float | Real | 32 | $3,4 \times 10^{-38}$ | $3,4 \times 10^{38}$ |
| double | Real | 64 | $1,7 \times 10^{-308}$ | $1,7 \times 10^{208}$ |

Nota: Valores referidos a arquitectura Intel

Variables en C

Tipos básicos de datos (II)

- Tabla referida a arquitectura Intel
 - Cada compilador/arquitectura interpreta estos valores
- El tipo `int` suele estar ajustado al tamaño de la palabra
- Existen varias notaciones para los números
 - Notación científica: `float var1=123.456E-7, var2=0.12E3;`
 - Notación octal: `int var=037;`
 - Notación hexadecimal: `int var=0x1F;`
 - Notación caracteres: `char var1='3', var2=0x33F;`
- Las variables se definen al principio de la función
- Ejemplo de declaración constante: `#define MAX 1000`
- Una variable no inicializada contiene basura, ¡cuidado!
- Existen tipos de datos no básicos: `enum`, `struct`

Operadores

Operadores aritméticos

Operadores aritméticos

| | |
|---|----------|
| + | Suma |
| - | Resta |
| * | Producto |
| / | División |
| % | Módulo |

- Con variables enteras y reales
- `*`, `/` y `%` preceden a `+` y `-`
 - $a * b + c \neq a * (b + c)$
- Pueden utilizarse paréntesis, de hecho, es recomendable

Ejemplo: Cálculo de un polinomio

```
int main() {
    float x=1.5, y;
    y = (3/2)*x*x + 1.2*x - 4
    printf("%f", y);
    return 0;
}
```

Operadores

Operadores relacionales y lógicos (I)

Operadores relacionales

== Igual
!= Distinto
> Mayor
>= Mayor o igual
< Menos
<= Menor o igual

Operadores relacionales

&& Operador AND
|| Operador OR
! Negación

- Resultado lógico: VERDADERO o FALSO
 - En C no existen variables lógicas
 - FALSE se representa como 0
 - VERDADERO se representa como distinto de 0
- Se usan mucho en bucles y condiciones
- El valor de las funciones lógicas se expresan con tablas de verdad

Tablas de verdad

| | | | | |
|------|---|---|---|---|
| A | T | T | F | F |
| B | T | F | T | F |
| A&&B | T | F | F | F |

| | | | | |
|------|---|---|---|---|
| A | T | T | F | F |
| B | T | F | T | F |
| A B | T | T | T | F |

Operadores

Operadores relacionales y lógicos (II)

Ejemplo 1

```
int numero;

printf("Escoja un entero entre el 10 y el 20:\n");
scanf("%d", &numero);

while((numero < 10) || (numero > 20))
    scanf("%d", &numero);
```

Ejemplo 2

```
int numero;
scanf("%d", &numero);
if ((numero > 10) && (numero < 20))
    printf("El entero es mayor que 10 y menor que 20");
```

Operadores

Operadores de incremento y decremento

Operadores relacionales

n++ Usar e incrementar
n-- Usar y decrementar
++n Incrementar y usar
--n Decrementar y usar

- ¡Sólo puede usarse con variables!
- Se usan mucho en bucles

Ejemplo

```
int n1 = 5, n2;  
int m1 = 5, m2;  
  
n2 = n1++;  
m2 = ++m2;  
  
printf("n1= %d, n2= %d\n",  
       n1, n2);  
printf("m1= %d, m2= %d\n",  
       n1, n2);
```

Operadores

Operadores a nivel de bit (I)

Operadores a nivel de bit

& AND a nivel de bit
| OR a nivel de bit
^ XOR a nivel de bit
« Rotación a izquierda
» Rotación a derecha
~ Inversión

AND

| | |
|-----|------|
| A | 1100 |
| B | 1010 |
| A&B | 1000 |

OR

| | |
|-----|------|
| A | 1100 |
| B | 1010 |
| A B | 1110 |

ROTACIÓN

| | |
|-----|------|
| A | 1100 |
| A»1 | 0110 |
| A»2 | 0011 |

XOR

| | |
|-----|------|
| A | 1100 |
| B | 1010 |
| A^B | 0110 |

- Actúan sobre cada bit por separado
 - Los operadores lógicos actúan sobre toda la variable
- Implementan operaciones lógicas clásicas
 - AND, OR, XOR, rotación e inversión
 - AND se usa para poner bits a 0
 - OR se usa para poner bits a 1

Operadores

Operadores relacionales y lógicos (II)

Ejemplo 1: Poner a 1

```
unsigned char a=0x0F, b=0xFF;
printf("a=%x, b=%x\n", a, b);
printf("a AND b = %x\n", a&b);
printf("a OR b = %x\n", a|b);
printf("a << 4 = %x\n", a<<4);
printf("~a = %x\n", ~a);
```

Salida ejemplo 1

```
a=f, b=ff
a AND b = f
a OR b = ff
a << 4 = f0
~a = ffffffff0
```

- Necesitamos operar sobre bits concretos: Máscara de bits
- Ejemplo: Obtener el valor del bit 5

```
xxx1xxxx &
00010000 =
-----
00010000
```

```
xxx0xxxx &
00010000 =
-----
00000000
```

Operadores

Operadores de asignación

Operadores de asignación

```
= Asignar
+= Sumar y asignar
-= Restar y asignar
*= Multiplicar y asignar
/= Dividir y asignar
```

La expresión

```
i = i + 2;
es equivalente a
i += 2;
```

```
condicion? si-verdadero :
si-falso;
```

- Las condiciones se usan constantemente
- El operador ternario permite simplificar el código

- ```
if (a>b)
 z = a;
else
 z = b;
```

- $$z = (a > b) ? a : b;$$

- Se puede usar en expresiones complejas

```
printf("El mayor es: %d", (a>b)? a : b);
```

- 1 Dado un número de tipo int, contar el número de "1" que tiene dicho número en binario.
- 2 Extraer los bits situados a partir de una determinada posición.

## Ejercicios

### Soluciones (I)

#### Solución ejercicio 1

```
#include <stdio.h>
#define N_BITS 32

int main() {
 int numero, i, mascara, unos=0;

 scanf("%d", &numero);
 for (i=0; i<N_BITS; i++) {
 if (i==0) mascara = 0x1;
 else mascara = mascara << 1;
 if ((numero & mascara) != 0) unos++;
 }
 printf("Resultado: %d\n", unos);
 return 0;
}
```



## Ejercicios

### Soluciones (II)

#### Solución ejercicio 2

```
#include <stdio.h>
#define N_BITS 32

int main() {
 int numero, desplazamiento, i, mascara=0x1;

 scanf("%d", &numero);
 scanf("%d", &desplazamiento);
 mascara = mascara << desplazamiento;

 for (i=0; i<N_BITS; i++) {
 printf(((numero & mascara) == 0) ? "0" : "1");
 mascara = mascara << 1;
 }
 return 0;
}
```

