Informática I

Claudio Paz

claudiojpaz@gmail.com

Octubre 2019

Unidad 11

Uso del lenguaje C en aplicaciones de bajo nivel

Operadores a nivel de bit

Operadores a nivel de bit

Como se vio en la primera unidad las computadoras representan toda la información interna usando *bits*.

Operadores a nivel de bit

Como se vio en la primera unidad las computadoras representan toda la información interna usando *bits*.

Los bits solo pueden tener dos valores, 0 y 1.

Operadores a nivel de bit

Como se vio en la primera unidad las computadoras representan toda la información interna usando *bits*.

Los bits solo pueden tener dos valores, 0 y 1.

Los operadores a nivel de bit realizan las operaciones *bit a bit* entre sus operandos

Operadores a nivel de bit

Como se vio en la primera unidad las computadoras representan toda la información interna usando *bits*.

Los bits solo pueden tener dos valores, 0 y 1.

Los operadores a nivel de bit realizan las operaciones *bit a bit* entre sus operandos

```
char a = 77;          // 0 1 0 0 1 1 0 1
char b = 42;          // 0 0 1 0 1 0 1 0
```

Operadores a nivel de bit

Como se vio en la primera unidad las computadoras representan toda la información interna usando *bits*.

Los bits solo pueden tener dos valores, 0 y 1.

Los operadores a nivel de bit realizan las operaciones *bit a bit* entre sus operandos

```
char a = 77;          // 0 1 0 0 1 1 0 1
char b = 42;          // 0 0 1 0 1 0 1 0
```

Operadores a nivel de bit

Como se vio en la primera unidad las computadoras representan toda la información interna usando *bits*.

Los bits solo pueden tener dos valores, 0 y 1.

Los operadores a nivel de bit realizan las operaciones *bit a bit* entre sus operandos

```
char a = 77;          // 0 1 0 0 1 1 0 1
char b = 42;          // 0 0 1 0 1 0 1 0
```

Operadores a nivel de bit

Como se vio en la primera unidad las computadoras representan toda la información interna usando *bits*.

Los bits solo pueden tener dos valores, 0 y 1.

Los operadores a nivel de bit realizan las operaciones *bit a bit* entre sus operandos

```
char a = 77;          // 0 1 0 0 1 1 0 1
char b = 42;          // 0 0 1 0 1 0 1 0
```

Operadores a nivel de bit

Como se vio en la primera unidad las computadoras representan toda la información interna usando *bits*.

Los bits solo pueden tener dos valores, 0 y 1.

Los operadores a nivel de bit realizan las operaciones *bit a bit* entre sus operandos

```
char a = 77;          // 0 1 0 0 1 1 0 1
char b = 42;          // 0 0 1 0 1 0 1 0
```

Operadores a nivel de bit

Como se vio en la primera unidad las computadoras representan toda la información interna usando *bits*.

Los bits solo pueden tener dos valores, 0 y 1.

Los operadores a nivel de bit realizan las operaciones *bit a bit* entre sus operandos

```
char a = 77;          // 0 1 0 0 1 1 0 1
char b = 42;          // 0 0 1 0 1 0 1 0
```

Operadores a nivel de bit

Como se vio en la primera unidad las computadoras representan toda la información interna usando *bits*.

Los bits solo pueden tener dos valores, 0 y 1.

Los operadores a nivel de bit realizan las operaciones *bit a bit* entre sus operandos

```
char a = 77;          // 0 1 0 0 1 1 0 1
char b = 42;          // 0 0 1 0 1 0 1 0
```

Operadores a nivel de bit

Como se vio en la primera unidad las computadoras representan toda la información interna usando *bits*.

Los bits solo pueden tener dos valores, 0 y 1.

Los operadores a nivel de bit realizan las operaciones *bit a bit* entre sus operandos

```
char a = 77;          // 0 1 0 0 1 1 0 1
char b = 42;          // 0 0 1 0 1 0 1 0
```

Operadores a nivel de bit

Como se vio en la primera unidad las computadoras representan toda la información interna usando *bits*.

Los bits solo pueden tener dos valores, 0 y 1.

Los operadores a nivel de bit realizan las operaciones *bit a bit* entre sus operandos

```
char a = 77;          // 0 1 0 0 1 1 0 1
char b = 42;          // 0 0 1 0 1 0 1 0
```

Operadores a nivel de bit. Operadores Lógicos

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: AND

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: AND

a_i	b_i	&
0	0	0
0	1	0
1	0	0
1	1	1

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: AND

a_i	b_i	&
0	0	0
0	1	0
1	0	0
1	1	1

```
a = 77;          // 0 1 0 0 1 1 0 1
b = 42;          // 0 0 1 0 1 0 1 0
c = a & b;       // 0 0 0 0 1 0 0 0
printf("%d\n", c);
```

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: AND

a_i	b_i	&
0	0	0
0	1	0
1	0	0
1	1	1

```
a = 77;          // 0 1 0 0 1 1 0 1
b = 42;          // 0 0 1 0 1 0 1 0
c = a & b;       // 0 0 0 0 1 0 0 0
printf("%d\n", c);
```

```
8
```

Operadores a nivel de bit. Operadores Lógicos

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: OR

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: OR

a_i	b_i	|
0	0	0
0	1	1
1	0	1
1	1	1

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: OR

a_i	b_i	|
0	0	0
0	1	1
1	0	1
1	1	1

```
a = 77;          // 0 1 0 0 1 1 0 1
b = 42;          // 0 0 1 0 1 0 1 0
c = a | b;       // 0 1 1 0 1 1 1 1
printf("%d\n", c);
```

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: OR

a_i	b_i	|
0	0	0
0	1	1
1	0	1
1	1	1

```
a = 77;          // 0 1 0 0 1 1 0 1
b = 42;          // 0 0 1 0 1 0 1 0
c = a | b;       // 0 1 1 0 1 1 1 1
printf("%d\n", c);
```

```
111
```

Operadores a nivel de bit. Operadores Lógicos

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: OR exclusiva (XOR)

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: OR exclusiva (XOR)

a_i	b_i	$\wedge$
0	0	0
0	1	1
1	0	1
1	1	0

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: OR exclusiva (XOR)

a_i	b_i	$\wedge$
0	0	0
0	1	1
1	0	1
1	1	0

```
a = 77;          // 0 1 0 0 1 1 0 1
b = 42;          // 0 0 1 0 1 0 1 0
c = a ^ b;       // 0 1 1 0 0 1 1 1
printf("%d\n", c);
```

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: OR exclusiva (XOR)

a_i	b_i	$\wedge$
0	0	0
0	1	1
1	0	1
1	1	0

```
a = 77;          // 0 1 0 0 1 1 0 1
b = 42;          // 0 0 1 0 1 0 1 0
c = a ^ b;       // 0 1 1 0 0 1 1 1
printf("%d\n", c);
```

```
103
```

Operadores a nivel de bit. Operadores Lógicos

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: Complemento

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: Complemento

a_i	$\sim$
0	1
1	0

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: Complemento

a_i	$\sim$
0	1
1	0

```
a = 77;          // 0 1 0 0 1 1 0 1
c = ~a;          // 1 0 1 1 0 0 1 0
printf("%d\n", c);
```

Operadores a nivel de bit. Operadores Lógicos

Tablas de verdad: Complemento

a_i	$\sim$
0	1
1	0

```
a = 77;          // 0 1 0 0 1 1 0 1
c = ~a;          // 1 0 1 1 0 0 1 0
printf("%d\n", c);
```

```
-78
```

Operadores de desplazamiento

Operadores de desplazamiento

Los operadores << y >> son operadores que generan un desplazamiento sobre los bits de un `int` o un `char`, con o sin signo.

Operadores de desplazamiento

Los operadores << y >> son operadores que generan un desplazamiento sobre los bits de un `int` o un `char`, con o sin signo.

El operando de la izq. es el que se ve afectado por el corrimiento.

Operadores de desplazamiento

Los operadores << y >> son operadores que generan un desplazamiento sobre los bits de un `int` o un `char`, con o sin signo.

El operando de la izq. es el que se ve afectado por el corrimiento.

El de la derecha indica cuantos bit de corrimiento.

Operadores de desplazamiento

Los operadores << y >> son operadores que generan un desplazamiento sobre los bits de un `int` o un `char`, con o sin signo.

El operando de la izq. es el que se ve afectado por el corrimiento.

El de la derecha indica cuantos bit de corrimiento.

```
a = 5;          // 0 0 0 0 0 1 0 1
c = a<<1;       // 0 0 0 0 1 0 1 0
printf("%d\n", c);
```

Operadores de desplazamiento

Los operadores << y >> son operadores que generan un desplazamiento sobre los bits de un `int` o un `char`, con o sin signo.

El operando de la izq. es el que se ve afectado por el corrimiento.

El de la derecha indica cuantos bit de corrimiento.

```
a = 5;          // 0 0 0 0 0 1 0 1
c = a<<1;       // 0 0 0 0 1 0 1 0
printf("%d\n", c);
```

```
10
```

Operadores de desplazamiento

Operadores de desplazamiento

Cuando se usa el operador << los bits se corren a la izquierda, agregando 0 (ceros) a la derecha.

Operadores de desplazamiento

Cuando se usa el operador << los bits se corren a la izquierda, agregando 0 (ceros) a la derecha.

Cuando se usa el operador >> los bits se corren a la derecha, agregando 0 (ceros) a la izquierda.

Operadores de desplazamiento

Cuando se usa el operador << los bits se corren a la izquierda, agregando 0 (ceros) a la derecha.

Cuando se usa el operador >> los bits se corren a la derecha, agregando 0 (ceros) a la izquierda. En caso de ser un número negativo pueden pasar dos cosas...Se agregan unos (el número sigue siendo negativo) o se agregan ceros (cambia el signo). Depende de la arquitectura.

Operadores a nivel de bit. Asignación

Operadores a nivel de bit. Asignación

&=	AND y asignación
|=	OR y asignación
^=	XOR y asignación
<<=	Desp. izq. y asignación
>>=	Desp. der. y asignación

Operadores. Precedencia

Operadores. Precedencia

Operador	Asociatividad
() [] . ->	Izq. a Der.
+ − (tipo) ++ −− ! ~ & *	Der. a Izq.
* / %	Izq. a Der.
+ −	Izq. a Der.
<< >>	Izq. a Der.
< <= > >=	Izq. a Der.
== !=	Izq. a Der.
&	Izq. a Der.
^	Izq. a Der.
|	Izq. a Der.
&&	Izq. a Der.
||	Izq. a Der.
? :	Der. a Izq.
= += −= /= *= &= |= ^= <<= >>= %=	Der. a Izq.
,	Izq. a Der.