Práctica de constantes, expresiones y operadores

Definición de variables (II)

- Las variables (*automáticas*) no inicializadas tienen cualquier valor.
- Una variable (*automática*) se inicializa cada vez que se llama a la función o bloque donde se declara.
- Las variables externas y estáticas se inicializan a cero por defecto.
- Se puede agregar el calificador **const** a la declaración de una variable para especificar que su valor no será modificado. Ejm:

```
const double e = 2.718281828459O5;
const char msg[ ] = "cuidado: ” ;
```

Constantes enumeradas

- Consisten de una lista de valores enteros constantes etiquetados.
- Sintaxis:

```
enum [identificador]{lista de enumerados}
```

- Ejemplos:

```
enum colores { rojo, azul, verde, amarillo }
enum {manzana=0, naranja=10, limon= -5, uva}
```

- El primer nombre la lista de enumerados tiene valor 0, el siguiente 1 y así sucesivamente, a menos que se asignen valores explícitos.
- Si no se especifican todos los valores, los valores sin especificar continuan con la progresión a partir del último valor especificado.

Ejemplos de constantes simbólicas

```
#define INTERES 0.16
#define PI 3.141593
#define TRUE 1
#define FALSE 0
#define AMIGO “Obama”
....
area = PI * radio * radio;
printf(“El interes es %f”, INTERES);
```

Ejemplo: Operadores aritméticos binarios

- Dadas las siguientes declaraciones:

```
int m = 3, n = 4;
float x = 2.5, y = 1.0;
```

Expresión	Expresión equiv.	Resultado
m + n + x + y		
m + n * x + y		
x / y + m / n		
x – y * m + y / n		
x / 0		

Operadores unarios + y –

Operador	Símbolo	Sintaxis	Operación
menos unario	–	– x	negación de x
más unario	+	+ x	valor del operando

- Ejm: j = 3 – –x

Operadores de asignación aritméticos

- Tienen menor precedencia y la asociatividad es de derecha a izquierda

Operador	Símbolo	Sintaxis	Operación
asignación	=	x = y	coloca valor de y en x
adición – asig.	+=	x += y	coloca valor de x+y en x
resta – asig.	–=	x –= y	coloca valor de x-y en x
multipl – asig.	*=	x *= y	coloca valor de x*y en x
división – asig.	/=	x /= y	coloca valor de x/y en x
resto – asig.	%=	x %= y	coloca valor de x%y en x

Ejemplo: Operadores de asignación aritméticos

- Dadas las siguientes declaraciones:

```
int m = 3, n = 4;
float x = 2.5, y = 1.0;
```

Expresión	Expresión equiv.	Resultado
m += n + x – y		
m /= n * x + y		
n %= (int) y + m		
x += y –= m		

Operadores de incremento, decremento

- Son operadores unarios que pueden ser prefijos o postfijos

Operador	Símbolo	Sintaxis	Operación
incr. postfijo	++	a++	obtiene valor de a, después incrementa a en 1
decr. postfijo	--	a--	obtiene valor de a, después decrementa a en 1
incr. prefijo	++	++a	incrementa a, después obtener valor de a en 1
decr. prefijo	--	--a	decrementa a, después obtener valor de a en 1

Ejemplo: Operadores de incremento, decremento

- Dadas las siguientes declaraciones:

```
int j = 0, m = 1, n= -1;
```

Expresión	Expresión equiv.	Resultado
m++ – --j m += ++j * 2 m++ * m++		

Operadores relacionales

- Se utilizan para formar expresiones lógicas que representan condiciones que pueden ser verdaderas o falsas. La expresión resultante es de tipo entero, ya que *verdadero* se representa por 1 y *falso* por 0 en C.

Operador	Símbolo	Sintaxis	Operación
mayor que	>	a > b	1 si a es mayor que b, sino 0
menor que	<	a < b	1 si a es menor que b, sino 0
mayor o igual a	>=	a >= b	1 si a es mayor o igual a b, sino 0
menor o igual a	<=	a <= b	1 si a es menor o igual a b, sino 0
igual a	==	a == b	1 si a es igual a b, sino 0
no igual a	!=	a != b	1 si a no es igual a b, sino 0

Ejemplo: Operadores relacionales

- Dadas las siguientes declaraciones:

```
int j=0, m=1, n=-1;
float x = 2.5, y = 0.0;
```

Expresión	Expresión equiv.	Resultado
j > m		
m / n < x		
j <= m >= n		
j <= x == m		
-x + j == y > n > m		
x += (y >=n)		
++ j == m != y * 2		

Operadores lógicos

- Actuan sobre operandos que son a su vez expresiones lógicas.

Operador	Símbolo	Sintaxis	Operación
AND lógico	&&	a && b	1 si a y b son 1, sino 0
OR lógico	||	a || b	1 si a o b son 1, sino 0
negación lógica	!	!a	1 si a es cero, sino 0

- Tener en cuenta la evaluación en corto circuito de los operadores relacionales para evitar efectos secundarios. Así en: `if ((a < b) && (c == d++))` d sólo se incrementa en caso que a sea menor que b.

Ejemplo: Operadores lógicos

- Dadas las siguientes declaraciones:

```
int j = 0, m = 1, n = -1;
float x = 2.5, y = 0.0;
```

Expresión	Expresión equiv.	Resultado
j && m		
j < m && n < m		
m + n || !j		
x * 5 && 5 || m / n		
!x || !n || m + n		
(j || m) + (x || ++ n)		

Operadores para manipulación de bits

- Se aplican sobre operandos de tipo integral. El operando izquierdo se promociona a un entero.

Operador	Símbolo	Sintaxis	Operación
desplazam. der.	>>	x >> y	x desplaz. der. y bits
desplazam. izq.	<<	x << y	x desplaz. izq. y bits
AND nivel bits	&	x & y	x ANDbits y
OR inclusivo bits	|	x | y	x ORbits y
OR exclusivo	^	x ^ y	x exORbits y
complemento bits	~	~x	complemento bits de x

Ejemplo: Operadores para manipulación de bits

Expresión	Expresión equiv.	Resultado
5 << 1	00000000 00000101	
	00000000 00001010	10
255 >> 3	00000000 11111111	
	00000000 00011111	31
9430 0x24D6	00100100 11010110	
5722 0x165A	00010110 01011010	
0x24D6 & 0x165A	00000100 01010010	0x0452
0x24D6 | 0x165A	00110110 11011110	0x35DE
0x24D6 ^ 0x165A	00110010 10001100	0x328C
~0x24D6	11011011 00101001	0xDB29

Operadores para asignación de bits

Operador	Símbolo	Sintaxis	Operación
asig. despl. der.	>>=	x >>= y	asigna x>>y a x
asig. despl. izq.	<<=	x <<= y	asigna x<<y a x
asigna - AND	&=	x &= y	asigna x & y a x
asigna - OR	|=	x |= y	asigna x | y a x
asigna - XOR	^=	x ^= y	asigna x ^ y a x

Operador condicional (ternario)

Operador	Símbolo	Sintaxis	Operación
condicional	? :	a ? b : c	si a no es cero devuelve b, sino c

Equivalencia:

```
if (x<y)                    z = (x < y) ? x : y ;
        z=x;
else
        z=y;
```

Ejm: `x = (a > 0) ? a : -a ;  /* valor absoluto de a */`

Operador paréntesis

- Permite agrupar los operandos y operadores que aparecen dentro de los paréntesis en primer lugar, alterando la precedencia natural de las operaciones.

Operador	Símbolo	Sintaxis	Operación
paréntesis	(…)	(a+b)/(c*d)	Altera la precedencia

Operador cast

- Permite convertir el valor resultante de una expersión a un tipo de datos diferente.

Operador	Símbolo	Sintaxis	Operación
cast	(tipo)	(tipo) expr.	Convierte expr. a tipo

- Ejm: (float) 3 / 2

Operador sizeof

- La llamada sizeof() se utiliza para determinar el número de bytes que ocupa una variable o un tipo

Operador	Símbolo	Sintaxis	Operación
sizeof	sizeof	sizeof (t) sizeof (x)	Devuelve el tamaño, en bytes, del tipo de dato t o expresión x.

- Ejm:

```
int a = 5;
printf ("bytes de a: %d\n",sizeof(a));
printf ("bytes de double: %d\n", sizeof(double));
```