Pre účely manipulácie s bytmi poskytuje jazyk C 6 operátorov:

& - bitový súčin (AND)

| - bitový súčet (OR)

^ - bitový exkluzívny súčet (XOR)

<< - posun doľava

>> - posun doprava

~ - jednotkový doplnok - negácia bit po bite - unárny operátor

bitový súčin  
***i-ty*** bit výsledku bitového súčinu:

x & y

bude ***1***, pokiaľ***i-ty*** bit***x*** a***i-ty*** bit***y*** budú ***1***, ináč bude ***0***.  
Teda jednotlivé bity výsledku budú záležať na jednotlivých bitoch operandov.  
  
**Príklad:**

#define je\_parne(x) (1 & (unsigned)(x))

bitový súčin sa často používa na vymaskovanie (nastavenie na nulu) určitých bitov,  
napr. ak chceme premennú typu int previesť na ASCII znak, teda využiť len najnižších 7 bitov:

c = c & 0x7F; /\* 0x7F je 0000 0000 1111 1111 \*/

alebo skrátene:  
**c &= 0x7F;**  
Poznámka:  
\* Uvedomme si, že je rozdiel medzi bitovým súčinom a logickým súčinom:

unsigned int i = 1, j = 2, k, l;

k = i && j; /\* k == 1 \*/

k = i & j; /\* k == 0 \*/

bitový súčet  
***i-ty*** bit výsledku bitového súčtu:

x | y

bude ***1***, pokiaľ***i-ty*** bit***x*** alebo ***i-ty*** bit ***y*** bude *1*,  
ak budú obidva nulové, bude výsledok***0***.  
bitový súčet sa často používa na nastavenie určitých bitov na ***1***, pričom sa ostatné bity  
nechajú nedotknuté.  
  
**Príklad:**  
Nasledujúce makro môže byť použité na zmenu veľkých písmen na malé:

#define na\_male( c ) (c | 0x20) /\* 0x20 je 0010 0000 binárne \*/

bitový exkluzívny súčet  
***i-ty*** bit výsledku bitového XOR:

x ^ y

bude ***1***, pokiaľ ***i-ty*** bit ***x*** sa nerovná ***i-temu*** bitu ***y***, ak sú obidva nulové, alebo obidva  
jednotkové bude výsledok ***0***.  
Táto operácia sa dá použiť k porovnaniu dvoch celých čísiel:

if (x ^ y)

/\* čísla sú rozdielne \*/

operácia bitového posunu doľava

x << n

Posunie bity v ***x*** doľava o ***n*** pozícií. Pri tomto posune sa zľava bity strácajú - sú vytlačované - a sprava sú doplňované ***0***. Bitový posun doľava sa občas používa na rýchle násobenie dvomi, respektívne mocninou dvoch. **Napr. Príkaz:**

x = x << 1;

alebo skrátene

x <<= 1;

vynásobí***x*** dvomi, alebo príkaz:

x <<= 3;

vynásobí ***x*** ôsmimi (8 = 23)

negácia bit po bite  
Na túto akciu sa tiež často používa názov jednotkový doplnok. **Príkaz:**

~x

Prevráti nulové bity na jednotkové a naopak. Tento operátor sa používa napr. v situáciách, keď sa chceme vyhnúť počítačovo závislej dĺžke celého čísla. **Napríklad príkaz:**

unsigned int x;

x &= 0xFFF0;

nastaví na ***nulu*** najnižšie štyri bity ***x***. Bude ale pracovať správne len na počítačoch,  
kde platí: sizeof(int) == 2.  
**Riešením je príkaz:**

x &= ~0xF;