

Bitové operace

napište kód, který nastaví n -tý bit v proměnné typu `unsigned int`. Následně napište kód, který ho vynuluje.

```
unsigned int vysledek = 0u;  
    // pro bitové operace používáme neznaménkový typ  
unsigned int pom = 1u;  
    // pouze pro demonstraci jednotlivých kroků - lze  
    // udělat v celku  
int n=4;  
    // který bit budeme nastavovat - bity jsou číslovány  
    // zprava a začínají bitem jedna
```

```
unsigned int vysledek = 0u;  
    // pro bitové operace používáme neznaménkový typ  
unsigned int pom = 1u;  
    // pouze pro demonstraci jednotlivých kroků - lze  
    // udělat v celku  
int n=4;  
    // který bit budeme nastavovat - bity jsou číslovány  
    // zprava a začínají bitem jedna  
  
pom = pom << (n-1);  
    // pomocnou jedničku posuneme o daný počet míst na  
    // požadovanou pozici  
  
vysledek = vysledek | pom;  
    // nastavení bitu pomocí operace bitový OR  
  
pom = ~pom;  
    // bitová negace - všude budou jedničky, pouze na  
    // zvoleném místě nula
```

```
vysledek &= pom;  
    // vynulování bitu pomocí operace bitový AND -  
    // „zkrácené“ přiřazení  
    // tj. vysledek = vysledek & pom;  
  
// nastavení bitu pomocí operace bitový OR -  
vysledek |= (1ul << (n-1));  
  
// "jednička" je typu long int (1ul), aby to fungovalo  
// pro všechny celočíselné typy (v C99 je ještě long long!)  
  
// změna bitu pomocí operace bitový Exclusive OR(XOR)-  
vysledek ^= (1ul << (n-1));
```

Napište podmínku pomocí bitových operací zjišťující, zda je celé číslo liché

```
int i;
```

```
...
```

```
if(i & 0x1) // poslední bit má váhu 1.
```

```
// ostatní bity reprezentují sudá čísla.
```

```
// je-li tedy nastaven poslední bit, je číslo liché
```

Napište podmínku pomocí bitových operací, zda je číslo dělitelné osmi.

```
#define MASKA_DELIT_8 0x07
```

```
int i;
```

```
// ptáme-li se na dělitelnost mocniny 2, je to obdobné,  
// jako s dělitelností 10 u desítkových čísel.
```

```
// bude dělitelné vždy, pokud od daného řádu do konce  
// budou nuly.
```

```
// Číslo osm je binárně 1000 tj. 0x8 hexadecimálně  
// Pro dělitelnost osmi tedy musí mít celé číslo na  
// posledních třech místech nuly
```

```
...  
if( (i & MASKA_DELIT_8) == 0 )
```

```
// Pro zjištění dělitelnosti osmi
```

```
// tedy na testované číslo použijeme masku, ve které jsou  
// nastaveny poslední tři bity.
```

```
// Pomocí operace AND a této masky vynulujeme všechny  
// bity dané proměnné a ponecháme pouze poslední tři bity  
// Je-li v těchto třech bitech nějaký nastaven, nebude  
// výsledek nula a číslo nebude dělitelné 8
```

Navrhněte strukturu bitově orientované proměnné, ve které je možné nastavit formát tisku celého čísla.

Způsob tisku soustavy: hexadecimálně, dekadicky, oktalově..

Způsob tisku znaménka: znaménko + se tiskne, znaménko + se netiskne.

Způsob tisku písmen: malá písmena, velká písmena.

Ukažte práci s danou proměnnou – nastavení nulování.

MSB 2^7			2^4	2^3	2^2	2^1	LSB 2^0
-	-	-	Oct	Dec	Hex	Znaménko	Velikost

```
#define TISK_VELIKOST 0x1  
#define TISK_ZNAMENKO 0x2  
#define TISK_SOUSTAVA 0x1C  
#define TISK_HEX 0x4  
#define TISK_DEC 0x8  
#define TISK_OCT 0x10
```

```
unsigned char tisk = TISK_DEC;
```

```
// přepnutí na hexa
```

```
tisk = tisk | TISK_HEX; // je to správně?
```

```
#define TISK_VELIKOST 0x1
#define TISK_ZNAMENKO 0x2
#define TISK_SOUSTAVA 0x1C
#define TISK_HEX 0x4
#define TISK_DEC 0x8
#define TISK_OCT 0x10
#define NASTAV(Val, Bit, Sekce) (((Val)&(~(Sekce)))|(Bit))
```

```
unsigned char tisk = TISK_DEC;
tisk = tisk | TISK_HEX; // přepnutí na hexa
// není správně, je nastaveno společně hex a dec
// správná varianta vyžaduje smazat všechny bity v daném
// poli a potom jeden nastavit
tisk = tisk & (~TISK_SOUSTAVA);
tisk |= TISK_HEX;
tisk = NASTAV(tisk, TISK_OCT, TISK_SOUSTAVA);
tisk |= TISK_ZNAMENKO;
//nastavení bitu tisku znaménka neovlivní bit soustavy
tisk &= ~TISK_ZNAMENKO; // vynulování bitu znaménka
```