Programozás C nyelven (12. ELŐADÁS)

Sapientia EMTE

2017-18

Bit-műveletek

Csak EGÉSZ típusokra alkalmazhatók: `char`, `int`

<<, >>, &, |, ^, ~

balra/ jobbra léptetés

ÉS, VAGY, kizáró-VAGY, NEM

A	B	A **XOR** B
0	0	0
0	1	1
1	0	1
1	1	0

Szemléltető PÉLDA (1)

```
unsigned char a=1, b=3;
```

Kifejezés	Belső ábrázolás	Érték
a	00000001	1
b	00000011	3
a << 3	00001000	8
a >> 1	00000000	0
a & b	00000001	1
a | b	00000011	3
a ^ b	00000010	2
~a	11111110	254

Szemléltető PÉLDA (2)

```
a * 2   ⇔   a << 1
a / 2   ⇔   a >> 1
```

```
char a=-1;
unsigned char b = 255;
```

Kifejezés	Belső ábrázolás	Érték
a	11111111	-1
b	11111111	255
a >> 1	11111111	-1
b >> 1	01111111	127
a << 1	11111110	-2
b << 1	11111110	254

Mi lesz az eredmény?

```
void xorSwap (int *x, int *y){
     if (x != y) {
          *x ^= *y;
          *y ^= *x;
          *x ^= *y;
     }
 }
```

Mi lesz az eredmény?

```
int main(void) {
    int arr[] = {12, 12, 14, 90, 14, 14, 14};
    int n = sizeof(arr)/sizeof(arr[0]);
    printf ("The … element is %d ", findOdd(arr, n));
    return 0;
}
```

```
int findOdd(int arr[], int n) {
    int res = 0, i;
    for (i = 0; i < n; i++)
      res ^= arr[i];
    return res;
}
```

Mi lesz az eredmény?

```
int main()
{
   int x = 19;
   printf ("x << 1 = %d\n", x << 1);
   printf ("x >> 1 = %d\n", x >> 1);
   return 0;
}
```

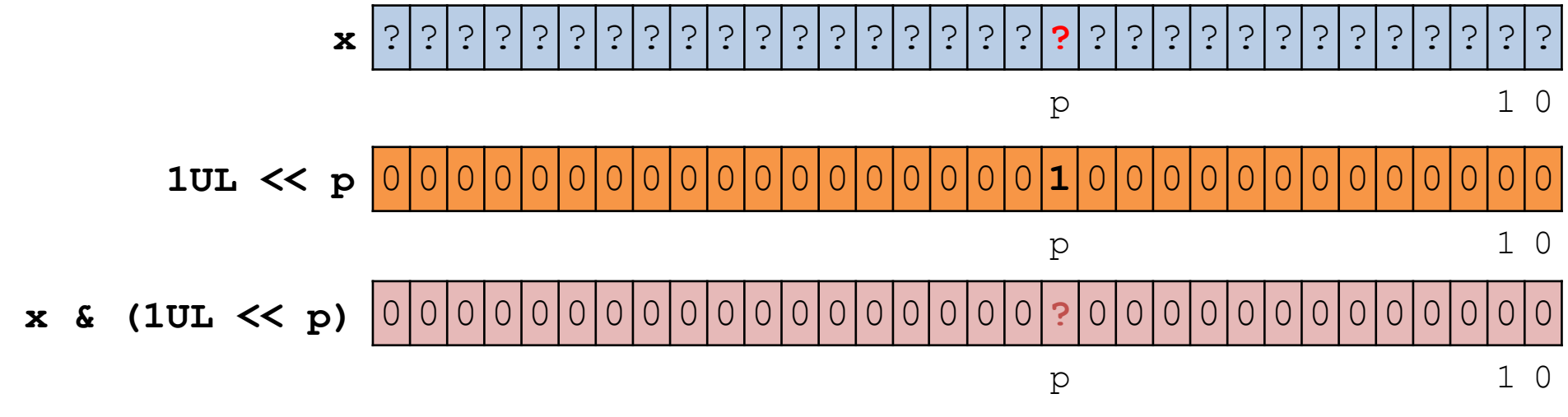
```
int main()
{
   int x = 2, y = 5;
   (x & y)? printf("True ") : printf("False ");
   (x && y)? printf("True ") : printf("False ");
   return 0;
}
```

Maszkolási technikák (1)

Ha b egy 0 vagy 1 értékű bit, akkor:

```
b | 0 = b
b | 1 = 1
b & 0 = 0
b & 1 = b
b ^ 0 = b
b ^ 1 = ~b
```

```
long x = 123456789;
int p = 13;
```



Maszkolási technikák (2)

Hogyan tudjuk lekérdezni egy egész változó p-edik helyértékű bitjét?

```
if (x & (1UL<<p)) ...
```

Hogyan tudjuk 0-ra állítani egy egész változó p-edik helyértékű bitjét?

```
x &= ~(1UL<<p);
```

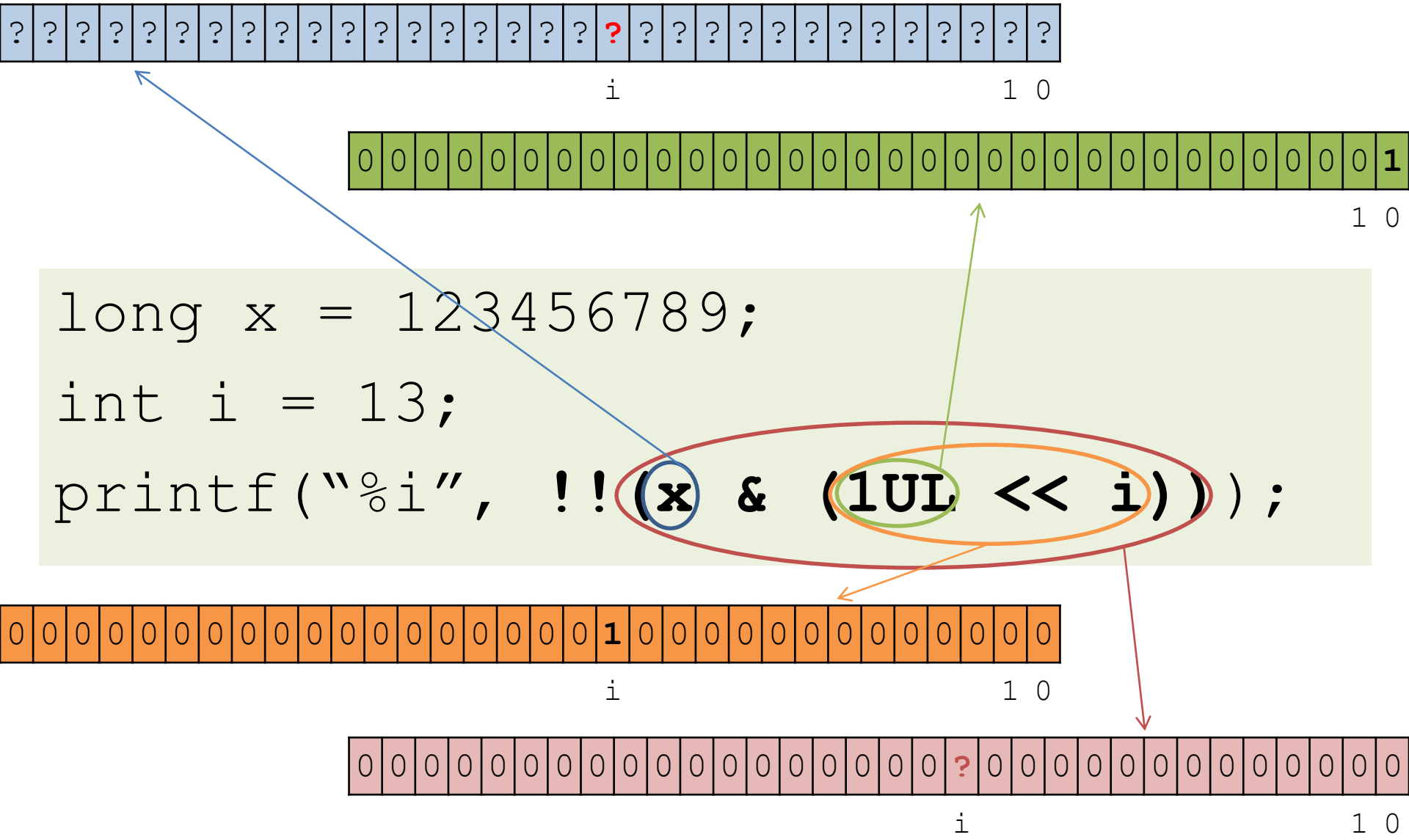
Hogyan tudjuk 1-re állítani egy egész változó p-edik helyértékű bitjét?

```
x |= 1UL<<p;
```

Hogyan tudjuk tagadni egy egész változó p-edik helyértékű bitjét?

```
x ^= 1UL<<p;
```

Maszkolási technikák (3)



5.1. feladat. Adott egy valós szám, amelyet `long double` változóban tárolunk el. Írjuk ki a belső ábrázolását.

```
long double x; int j;
unsigned char maszk;
char *p = (char*)&x; p += sizeof(long double)-1;
/* Definiálunk egy char-pointert, amit ráállítunk
    x legnagyobb helyértékű byte-jára */
scanf("%Lf", &x);
for( j = 1; j <= sizeof(long double); j++)
{
    maszk = 1 << sizeof(char)*8-1;
    while(maszk)
    {
        if(*p & maszk)putchar('1');
        else putchar('0');
        maszk >>= 1;
    }
    p--;
}
```

Összefoglalás

- Csak egész típusokra alkalmazhatók
  - Balra/jobbra tolás: **<<, >>**
  - ÉS, VAGY, KIZÁRÓ-VAGY: **&, |, ^**
  - Tagadás: **~**
- Ne kavarjuk össze a logikai ÉS/VAGY, illetve a bitenkénti ÉS/VAGY műveleteket
- A bit-műveletek alkalmazása hatékony kódot eredményez (példa: 2^n…)
  - `for( i = p = 1 ; i <= n ; ++i ){ p *= 2; }`
  - `p = 1 << n;`
- Ha valós értékekre szeretnénk bit műveleteket alkalmazni, akkor pointerekkel…

```
float x = 3.14;
long *p = (long*)(&x);
(*p) <<= 13;
```