Προγραμματισμός Ι

Χειρισμός Bits

Κωνσταντίνος Τσερπές

(βασισμένο στις διαφάνειες του κ. Δημήτρη Μιχαήλ)

Τμήμα Πληροφορικής και Τηλεματικής
Χαροκόπειο Πανεπιστήμιο

Bits

Οι υπολογιστές χρησιμοποιούν το δυαδικό σύστημα για να αναπαραστήσουν πληροφορία.

Η γλώσσα C έχει την δυνατότητα να χειριστεί σε λεπτομέρεια τα bits της αναπαράστασης των διαφόρων τύπων.

Στην ANSI C ο ακέραιος πρέπει να είναι τουλάχιστον 16 bits, αλλά στις σύγχρονες αρχιτεκτονικές ο ακέραιος είναι συνήθως 32 bits.

00000000 00000000 00000000 00000000

Στην C μπορείτε να δείτε το μέγεθος ενός τύπου σε *bytes* με

```
sizeof ( int )
```

Bitwise Τελεστές

Η C παρέχει τελεστές *bitwise* για να χειριστεί τα bits των τύπων:

1. char
2. short
3. int
4. long

και σε signed και σε unsigned μορφή.

Bitwise Τελεστές

Οι *bitwise* τελεστές που μας παρέχει η C είναι:

Σύζευξη επιπέδου bit (bitwise AND)	&
Διάζευξη επιπέδου bit (bitwise OR)	|
Αποκλειστική διάζευξη επιπέδου bit (bitwise XOR)	^
Συμπλήρωμα ως προς 1	~
Ολίσθηση κατά *n* bits αριστερά	<<
Ολίσθηση κατά *n* bits δεξιά	>>

Bitwise Τελεστής AND

Ο τελεστής AND (&) κάνει λογικό AND μεταξύ των τελεστέων bit με bit.

```
1  int  a , b , c ;
2
3  a = 548884;
4  b = 1595404;
5  c = a & b;
6  //  c now is       540676
```

```
00000000  00001000  01100000  00010100
                  &
00000000  00011000  01011000000001100
                  =
00000000  00001000  01000000000000100
```

Bitwise Τελεστής OR

Ο τελεστής OR (|) κάνει λογικό OR μεταξύ των τελεστέων bit με bit.

```
1  int  a , b , c ;
2
3  a = 548884;
4  b = 1595404;
5  c = a | b;
6  //  c now is       1603596
```

00000000 00001000 01100000 00010100

|

00000000 00011000 01011000000001100

=

00000000 00011000 01111000000001100

Bitwise Τελεστής Αποκλειστικού OR (XOR)

Ο τελεστής XOR (^) κάνει λογικό XOR μεταξύ των τελεστέων bit με bit.

```
1  int  a , b , c ;
2
3  a = 548884;
4  b = 1595404;
5  c = a ^ b;
6  //  c now is       1062936
```

00000000 00001000 01100000 00010100

xor

00000000 00011000 01011000 00001100

=

00000000 00010000 00111000 00011000

Αριστερή Μετατόπιση (<<)

Μετατοπίζει τα bits προς τα αριστερά και γεμίζει με 0 τα bits στα δεξιά.
Αυτό είναι το ίδιο με πολλαπλασιασμό με δύο.

```
1  int  a , c ;
2
3  a = 548884;
4  c = a << 10;
5   //  c is      562057216 same as  a          2^10
```

00000000 00001000 01100000 00010100

<<

10

=

00100001 10000000 01010000 00000000

Δεξιά Μετατόπιση (>>)

Μετατοπίζει τα bits προς τα δεξιά. Εαν ο τύπος είναι unsigned τότε γεμίζει τα bits αριστερά με 0. Αλλιώς τα νέα bits διατηρούν το πρόσημο, 0 ή 1 ανάλογα με την αρχιτεκτονική του συστήματος.

```
1  int  a , c ;
2
3  a = 548884;
4  c = a >> 10;
5  //  c is      536
```

00000000 00001000 01100000 00010100

>>

10

=

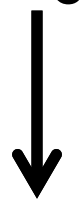
00000000 00000000 00000010 00011000

Συμπλήρωμα

Ο τελεστής συμπληρώματος (με βάση το 1) αντιστρέφει όλα τα bits.

```
1  int  a ,  c ;
2  a = 2032;
3  c = ~a;
4  // c is      -2033
```

00000000 00000000 00000111 11110000



11111111 11111111 11111000 00001111

Λογικοί Τελεστές και Τελεστές bit

Οι λογικοί τελεστές δεν έχουν καμία σχέση με τους τελεστές bit.

- 9 && 6 ⇒ true
- 9 & 6 ⇒ false
- !1 ⇒ false
- ~1 ⇒ true (-2)

Στην εντολή ελέγχου if/else πάντα χρησιμοποιούμε τους λογικούς τελεστές.

Προτεραιότητα Τελεστών

() [] . -> **expr++ expr--**	αριστερά προς τα δεξιά	υψηλότεροι
+ - ++**expr** --**expr** ! & * sizeof	δεξιά προς τα αριστερά	μοναδιαίοι
*/%	αριστερά προς τα δεξιά	πολλαπλασ.
+-	αριστερά προς τα δεξιά	προσθετικοί
<< >>	αριστερά προς τα δεξιά	μετατόπισης
< <= > >=	αριστερά προς τα δεξιά	σχεσιακοί
== !=	αριστερά προς τα δεξιά	ισότητας
&	αριστερά προς τα δεξιά	bitwise AND
^	αριστερά προς τα δεξιά	bitwise XOR
|	αριστερά προς τα δεξιά	bitwise OR
&&	αριστερά προς τα δεξιά	λογικό AND
||	αριστερά προς τα δεξιά	λογικό OR
?:	δεξιά προς τα αριστερά	συνθηκών
= += -= *= /= &= |= ^= <<= >>= %=	δεξιά προς τα αριστερά	εκχώρησης
,	αριστερά προς τα δεξιά	κόμμα

Προσοχή στην Προτεραιότητα

Η προτεραιότητα των τελεστών bit είναι χαμηλότερη από αυτών της σύγκρισης.

```
if ( value & 0x0f == 0 )        { /* ...      */ }
```

είναι ισοδύναμο με

```
if ( value & ( 0 x0f == 0 )        { /* ...      */ }
```

αντί αυτού που θέλαμε

```
if ( ( value & 0x0f ) == 0 )        { /* ...      */ }
```

Προσοχή στην Προτεραιότητα

Οι τελεστές ολίσθησης έχουν χαμηλότερη προτεραιότητα από τους αριθμητικούς τελεστές.

```
x = i << 2 + 1;
```

είναι ισοδύναμο με

```
x = i << (2 + 1);
```

αντί αυτού που θέλαμε

```
x = ( i << 2 ) + 1;
```

Τελεστής Ανάθεσης

Οι τελεστές κατά bit μπορούν να συνδυαστούν και με τον τελεστή ανάθεσης όπως οποιοδήποτε άλλος τελεστής (π.χ. αριθμητικός)

```
1  int a , b , c , d ;
2
3  a &= 1 ;
4  b |= 100;
5  b ^= a ;
6  c <<= 2;
7  d >>= 4;
```

Παράδειγμα

Εκτύπωση δυαδικής αναπαράστασης

```
1  #include  <stdio.h>
2
3  int  main ( )
4  {
5        int  x  =  2032 ,  i ;
6
7        for (  i  =      31; i     > −1; i−− )
8              printf ( "%d", (x & (1<< i))?1:0);
9
10       printf ( "\n" ) ;
11 }
```