Μάθημα 22:

Πράξεις bit

Δημήτρης Ψούνης



Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

Περιεχόμενα Μαθήματος

A. Bits & Bytes

- 1. Εισαγωγή
 - 1. Αποδομόντας μία Μεταβλητή
 - Συνάρτηση εξαγωγής bit
- 2. Τελεστές Ολίσθησης
 - 1. Αριστερή ολίσθηση
 - 2. Δεξιά ολίσθηση
- 3. Λογικοί Τελεστές (bitwise)
 - Λογικό ΚΑΙ (&)
 - Λογικό Ή (|)
 - Λογικό Αποκλειστικό ή (^)
 - Λογικό **ΟΧΙ** (~)
 - Συνάρτηση set_bit
- 4. Δομές και μεταβλητές bit
 - 1. Συντακτικό bit σε δομές.
 - 2. Παράδειγμα
- Β. Ασκήσεις

Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit



A. Bits & Bytes

1. Εισαγωγή

1. Αποδομόντας μία μεταβλητή

- Γνωρίζουμε ότι οι τύποι δεδομένων στην C δεσμεύουν κάποιο προκαθορισμένο πλήθος bytes.
- Όπως είδαμε στο μάθημα 3 «Μεταβλητές και Σταθερές»
 - Ένας char ακέραιος δεσμεύει 1 byte
 - Ένας short ακέραιος δεσμεύει 2 bytes
 - Ένας int ακέραιος συνήθως δεσμεύει 4 bytes
 - Ένας long ακέραιος δεσμεύει 8 bytes
- Και γνωρίζουμε ότι κάθε ακέραιος έχει δύο εκδοχές
 - Την προσημασμένη (μπορούμε να αποθηκεύσουμε αρνητικές και θετικές τιμές)
 - Την μη προσημασμένη (μπορούμε να αποθηκεύσουμε μόνο θετικές τιμές)
- Υπενθυμίζουμε (επόμενη διαφάνεια) και τα εύρη τιμών των τύπων αυτών

Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

A. Bits & Bytes

1. Εισαγωγή

1. Αποδομόντας μία μεταβλητή

Όνομα Τύπου Δεδομένων	Συμβολισμός	bytes	Εύρος τιμών
Χαρακτήρας	char	1	-128 εώς 127
Μικρός Ακέραιος	short	2	-32768 εώς 32767
Ακέραιος	int	4	-2147483648 εώς 2147438647
Μεγάλος Ακέραιος	long	4	-2147483648 εώς 2147438647

Όνομα Τύπου Δεδομένων	Συμβολισμός	bytes	Εύρος τιμών
Μη προσημασμένος Χαρακτήρας	unsigned char	1	0 εώς 255
Μη προσημασμένος Μικρός Ακέραιος	unsigned short	2	0 εώς 65535
Μη προσημασμένος Ακέραιος	unsigned int	4	0 εώς 4294967295
Μη προσημασμένος Μεγάλος Ακέραιος	unsigned long	4	0 εώς 4294967295

A. Bits & Bytes

1. Εισαγωγή

1. Αποδομόντας μία μεταβλητή

- Γνωρίζουμε ότι:
 - Κάθε byte έχει 8 bits
 - Κάθε αριθμός αποθηκεύεται στο δυαδικό σύστημα αρίθμησης.
- Οπότε μπορούμε να βρούμε τη δυαδική αναπάράσταση ενός ακεραίου, αποδομώντας τον αριθμό στα αντίστοιχα bits.
- Στο πρόγραμμα της επόμενης διαφάνειας, αποδομούμε έναν unsigned char (ο απλούστερος μη προσημασμένος ακέραιος) στα αντίστοιχα bits τα οποία περιέχει:

Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

www.psounis.gr

A. Bits & Bytes

1. Εισαγωγή

1. Αποδομόντας μία μεταβλητή

- Δεδομένου του unsigned char byte (π.χ. 3)
- Η δυαδική του αναπαράσταση είναι:

0 0 0 0 0 0 1 1

• Η πράξη (υπόλοιπο διαίρεσης με το 2):

byte%2

- απομονώνει το τελευταίο bit.
- ενώ η πράξη (ακέραια διαίρεση με το 2):

byte/2

• μετακινεί τα bits του αριθμού μία θέση δεξιά:

0 0 0 0 0 0 0 1

Για υπενθύμιση στην δυαδική αναπαράσταση αριθμών, βλέπε «Εισαγωγή στους Η/Υ – Συστήματα Αρίθμησης»

Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

A. Bits & Bytes

1. Εισαγωγή

1. Αποδομόντας μία μεταβλητή

/* bits_in_char.c Bits σε έναν μη προσημασμένο χαρακτήρα */
#include <stdio.h>

#define BITS 8

int main()
{
 unsigned char byte = 3;
 int bits[BITS];
 int i;

for (i=0; i<BITS; i++)
 {
 bits[BITS-1-i]=byte%2;
 byte=byte/2;
 }

for (i=0; i<BITS; i++)
 printf("%d", bits[i]);

return 0;
}

Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

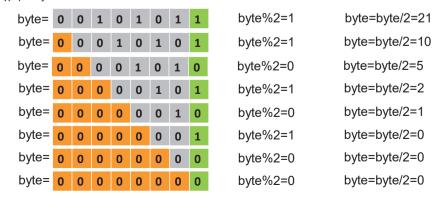
www.psounis.gr

A. Bits & Bytes

1. Εισαγωγή

1. Αποδομόντας μία μεταβλητή

- Με βάση τα παραπάνω, η επανάληψη αυτών των πράξεων απομονώνει ένα ένα τα bits του αριθμού.
 - Σύμφωνα με τον αλγόριθμο μετατροπής ενός δεκαδικού στον αντίστοιχο δυαδικό.
- Π.χ. για byte=43



• Και αποθηκεύουμε τα bytes με αντίστροφη σειρά εξαγωγής: (43)₁₀ = (00101011)₂

A. Bits & Bytes

Εισανωνή

2. Συνάρτηση εξαγωγής του bit σε μία θέση

- Γενικεύοντας το συλλογισμό μπορούμε να κατασκευάσουμε τις εξής συναρτήσεις:
- Τη συνάρτηση get bit που επιστρέφει το bit που είναι στην i-οστή θέση του byte:

```
int get bit(int pos, unsigned char byte char)
  int i:
  for (i=0; i<BITS-pos-1; i++)
     byte char=byte char/2:
  return byte char%2;
```

Και τη συνάρτηση εκτύπωσης που τυπώνει ένα byte

```
void bits_print(unsigned char byte)
 int i;
  for (i=0; i<BITS; i++)
     printf("%d", get_bit(i,byte));
```

Ολοκληρώμένο το πρόγραμμα είναι το bits_get_function.c

Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

A. Bits & Bytes

2. Τελεστές Ολίσθησης

2. Δεξιά ολίσθηση

- Με τον όρο δεξιά ολίσθηση (right shift), εννοούμε:
 - Την μετακίνηση μία θέση δεξιά των bits
 - Το δεξιότερο bit (καλείται και λιγότερο σημαντικό) χάνεται.
 - Το αριστερότερο bit (καλείται και πιο σημαντικό) γίνεται 0.
- Παράδειγμα:
 - Αρχικό byte:
 - 0 1 0 1 0 1 • Μετά από δεξιά ολίσθηση 0 0 0 1 0 1 0 1
- Παρατηρήσεις:
 - Ισοδυναμεί με ακέραια διαίρεση του αριθμού που απεικονίζεται επί δύο
- Ο τελεστής δεξιάς ολίσθησης στη C είναι ο >>
- και χρησιμοποιείται ως εξής:

var >> n

- όπου var μια μεταβλητή (π.χ. unsigned char)
- και η πόσες δεξιές ολισθήσεις θα γίνουν.

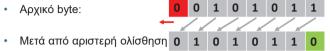
Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

A. Bits & Bytes

2. Τελεστές Ολίσθησης

1. Αριστερή ολίσθηση

- Με τον όρο αριστερή ολίσθηση (left shift), εννοούμε:
 - Την μετακίνηση μία θέση αριστερά των bits
 - Το αριστερότερο (καλείται και πιο σημαντικό) bit χάνεται.
 - Το δεξιότερο bit (καλείται και λιγότερο σημαντικό) γίνεται 0.
- Παράδεινμα:
 - Apylkó byte:



- Παρατηρήσεις:
 - Ισοδυναμεί με πολλαπλασιασμό του αριθμού που απεικονίζεται επί δύο
 - Και ακόμη ακριβέστερα (αν B είναι το byte) με την πράξη: B = (2 * B)%28
- Ο τελεστής αριστερής ολίσθησης στη C είναι ο <<
- και χρησιμοποιείται ως εξής:

var << n

- όπου var μια μεταβλητή (π.χ. unsigned char)
- και η πόσες αριστερές ολοσθήσεις θα γίνουν.

Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit



A. Bits & Bytes

2. Τελεστές Ολίσθησης

2. Δεξιά ολίσθηση

Με χρήση της αριστερής ολίσθησης μπορούμε να απλοποιήσουμε τη συνάρτηση εξαγωγής bit

```
int get bit(int pos, unsigned char byte)
   return (byte >> BITS-1-pos)%2;
```

Π.χ. αν θέλουμε να πάρουμε την 2^η θέση (pos=2):



• Πρέπει να κάνουμε 5 ολισθήσεις (BITS-1-pos)



• και στη συνέχεια κάνοντας mod 2 παίρνουμε το τελευταίο bit

Ολοκληρώμενο το πρόγραμμα είναι το bits get function shift.c

• Οι τελεστές << και >> δουλεύουν μόνο με ακεραίους: short, int, long και όχι float ή double

www.psounis.gr

A. Bits & Bytes

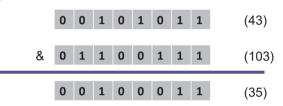
3. Λογικοί Τελεστές (bitwise)

Λογικό ΚΑΙ (&)

 Ο τελεστής λογικό ΚΑΙ (&) εκτελεί τη λογική πράξη ΑΝD μεταξύ των αντιστοίχων bits δύο αριθμών:

X	Υ	AND
0	0	0
0	1	0
1	0	0
1	1	1

• Παράδειγμα:



Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

www.psounis.g

A. Bits & Bytes

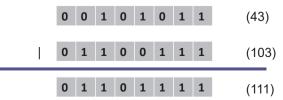
3. Λογικοί Τελεστές (bitwise)

2. Λογικό ή (|)

• Ο τελεστής λογικό ΚΑΙ (&) εκτελεί τη λογική πράξη ΑΝD μεταξύ των αντιστοίχων bits δύο αριθμών:

X	Υ	OR
0	0	0
0	1	1
1	0	1
1	1	1

• Παράδειγμα:



Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

A. Bits & Bytes

3. Λογικοί Τελεστές (bitwise)

1. Λογικό KAI (&)

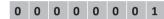
- Το λογικό ΚΑΙ μπορεί να χρησιμοποιηθεί για να κάνουμε την ονομαζόμενη «μάσκα» και να απομονώσουμε ένα bit ενός αριθμού.
- Π.χ. αν θέλουμε να πάρουμε την 2η θέση (pos=2):

1 0 1 0 1 0 1 1

• Πρέπει να κάνουμε 5 ολισθήσεις (BITS-1-pos)



• και στη συνέχεια κάνοντας μάσκα με το 1



• παίρνουμε το τελευταίο bit

• Συνεπώς αλλάζουμε την συνάρτηση ως εξής:

```
int get_bit(int pos, unsigned char byte)
{
    return (byte >> BITS-1-pos) & 1;
}
```

Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit



A. Bits & Bytes

3. Λογικοί Τελεστές (bitwise)

2. Λογικό ή ()

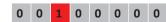
- Το λογικό ή μπορεί να χρησιμοποιηθεί για να θέσουμε ένα bit σε 1
- Π.χ. για να αλλάξουμε το 2ο bit

1 0 0 0 1 0 1 1

• Ξεκινάμε μάσκα με το 1



• Το φέρνουμε στη σωστή θέση (BITS-1-POS) αριστερές ολοσθήσεις



• Και το λογικό ή θα δώσει το αποτέλεσμα που θέλουμε:

```
1 0 1 0 1 0 1 1
```

• Γράφουμε μερικώς τη συνάρτηση:

```
void set_bit(int pos, unsigned char *byte, int val)
{
    unsigned char mask = 1;
    if (val==1) *byte = (mask << BITS-1-pos) | *byte;
```



A. Bits & Bytes

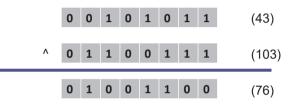
3. Λογικοί Τελεστές (bitwise)

3. Λογικό Αποκλειστικό ή (^)

• Ο τελεστής λογικό ΑΠΟΚΛΕΙΣΤΙΚΟ ή (^) εκτελεί τη λογική πράξη ΧΟR μεταξύ των αντιστοίχων bits δύο αριθμών:

X	Υ	XOR
0	0	0
0	1	1
1	0	1
1	1	0

• Παράδειγμα:



A Dita & Dutas

A. Bits & Bytes

3. Λογικοί Τελεστές (bitwise)

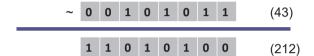
Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

4. Λογικό ΟΧΙ (~)

• Ο τελεστής λογικό ΟΧΙ (~) εκτελεί τη λογική πράξη ΝΟΤ επί ενός αριθμού:

X	XOR
0	1
1	0

Παράδειγμα:



Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

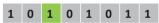


A. Bits & Bytes

3. Λογικοί Τελεστές (bitwise)

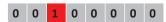
4. Λονικό ΟΧΙ (~)

- Με το NOT μπορούμε να κατασκευάσουμια ενδιαφέρουσα μάσκα για να θέσουμε ένα bit σε 0.
- Π.χ. για να αλλάξουμε το 2ο bit



• Ξεκινάμε τη μάσκα με το 1

• Το φέρνουμε στη σωστή θέση (BITS-1-POS) αριστερές ολοσθήσεις



• Αντιστρέφουμε τη μάσκα:



• Και κάνοντας λογικό ΚΑΙ με το αρχικό byte κάναμε το επιθυμητό:

• Σε κώδικα:

*byte = ~(mask << BITS-1-pos) & *byte;

A D'1 0 D 1

A. Bits & Bytes

3. Λογικοί Τελεστές (bitwise)

Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

5. Συνάρτηση set bit

• Συνεπώς κατασκευάσαμε τη συνάρτηση:

```
void set_bit(unsigned char *byte, int pos, int val)
{
  unsigned char mask = 1;

  if (val==0)
     *byte = ~(mask << BITS-1-pos) & *byte;
  else
     *byte = (mask << BITS-1-pos) | *byte;
}</pre>
```

• Μέσω της οποίας μπορούμε να κάνουμε ενέργειες όπως:

```
int main()
{
   unsigned char byte = 53;
   set_bit(&byte,2,1);

   return 0;
}
```

Ολοκληρώμένο το πρόγραμμα είναι το bits_set_function.c

A. Bits & Bytes

4. Δομές και μεταβλητές bit

- 1. Συντακτικό bit σε δομές.
- Στις δομές (structs) μπορούμε να ορίσουμε πεδία που είναι bits.
- Γίνεται με το εξής συντακτικό:

```
struct name
{
  type bit_name1: n1;
  type bit_name2: n2;
  ...
}
```

- όπου ορίζουμε πρώτα:
 - Σε τι μεταβλητές θα μεταφράζονται τα bits (είναι int ή unsigned int)
- έπειτα δίνουμε ένα όνομα
- και μετά την άνω κάτω τελεία
 - ορίζουμε το πλήθος των bits που θα χρησιμοποιήσει.
- Μπορούμε να χρησιμοποιήσουμε όσα τέτοια πεδία θέλουμε σε μία δομή.

Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit



A. Bits & Bytes

4. Δομές και μεταβλητές bit

2. Παράδειγμα

```
/* bits_in_struct.c Δομή που περιέχει πεδία bit */
#include <stdio.h>
#include <string.h>

struct student
{
    unsigned int active: 1;
    unsigned int type: 2;
    char name[80];
};

main()
{
    struct student bob;

    bob.active=0;
    bob.type=2;
    strcpy(bob.name, "Bob");

    printf("%d %d %s", bob.active, bob.type, bob.name);
}
```

Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

A. Bits & Bytes

4. Δομές και μεταβλητές bit

2. Παράδειγμα

- Για παράδειγμα έστω ότι κατασκευάζουμε μία δομή «Φοιτητής» η οποία:
 - Έχει ένα πεδίο «ενεργός» που είναι 0 ή 1
 - Έχει ένα πεδίο «είδος φοίτησης» που είναι 0 (προπτυχιακός), 1(μεταπτυχιακός), 2(διδακτορικός)
 - και συνηθισμένα πεδία όπως όνομα, επώνυμο κ.λπ.
- Ορίζουμε τη δομή με τον εξής τρόπο:

```
struct student
{
  unsigned int active: 1;
  unsigned int type: 2;
  char name[80];
  ...
}
```

- Στην μεταβλητή active μπορούμε να βάζουμε τιμή 0 ή 1
- Στην μεταβλητή type μπορούμε να βάζουμε τιμή 0 ή 1 ή 2
 - Αφού χρησιμοποιούμε 2 bits μπορούμε να αποθηκεύουμε 4 αριθμούς (0,1,2,3)

Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit



A. Bits & Bytes

5. Συναρτήσεις διαχείρισης μνήμης

1. Η συνάρτηση memset

- Προσφέρονται 3 συναρτήσεις οι οποίες διαχειρίζονται απ'ευθείας bytes:
- Η συνάρτηση memset:

```
void *memset(void *dest, int c, size_t count);
```

- και έχει οριστεί στο stdio.h
- Η συνάρτηση θέτει στα πρώτα count bye του dest το χαρακτήρα c.
- Παράδειγμα: Αν str="xxxxxxxxxx", η εντολή:

memset(str, 'a', 4)

• θα μετατρέψει το str="aaaaxxxxxx"

A. Bits & Bytes

5. Συναρτήσεις διαχείρισης μνήμης

2. Η συνάρτηση memcpy

• Η συνάρτηση memcpy:

void *memcpy(void *dest, void *src, size t count)

- και έχει οριστεί στο stdio.h
- Η συνάρτηση αντιγράφει count bytes από το src στο dest.
- Αν οι δύο χώροι μνήμης επικαλύπτονται ενδέχεται να αποτύχει.
- Παράδειγμα: Αν η str1="xxxxxxxxxxx" και η str2="aaaa", τότε η εντολή:

memset(str1, str2, 3)

- θα μετατρέψει το str1="aaaxxxxxxx"
- Παράδειγμα επικαλυπτόμενης μνήμης που η memcpy μπορεί να αποτύχει (π.χ. για count=4):



A. Bits & Bytes

5. Συναρτήσεις διαχείρισης μνήμης

2. Η συνάρτηση memmove

• Η συνάρτηση memmove:

void *memmove(void *dest, void *src, size t count)

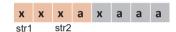
• και έχει οριστεί στο stdio.h

Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

- Η συνάρτηση αντιγράφει count bytes από το src στο dest.
- Αν οι δύο χώροι μνήμης επικαλύπτονται κάνει την αναμενόμενη ενέργεια. Στο προηνουμένο παράδειγμα:



- Η εντολή memset(str1, str2, 4)
- Θα δώσει τη συμβολοσειρά



Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit

__www.psounis.gr

<u>Β. Ασκήσεις</u> 1. LSB & MSB

Γράψτε μία συνάρτηση int LSB(unsigned char byte) η οποία

• Θα επιστρέφει το λιγότερο σημαντικό ψηφίο (δεξιότερο bit) ενός byte

Γράψτε μία συνάρτηση int MSB(unsigned char byte) η οποία

• Θα επιστρέφει το πιο σημαντικό ψηφίο (αριστερότερο bit) ενός byte

Επαληθεύστε την ορθότητα των δύο συναρτήσεων μέσω κατάλληλης συνάρτησης main

Δημήτρης Ψούνης, Η Γλώσσα C, Μάθημα 22: Πράξεις bit



Β. Ασκήσεις

2. Δεξιά περιστροφή

Γράψτε μία συνάρτηση void rotate(unsigned char *byte) η οποία

- Θα δέχεται ένα byte και θα μετακινεί κυκλικά κατά μία θέση δεξιά όλα τα bit του αριθμού
- Π.χ. το 00001111 θα πρέπει να μετατραπεί με μία δεξιά κυκλική ολίσθηση στο 10000111.

Επαληθεύστε την ορθότητα των δύο συναρτήσεων μέσω κατάλληλης συνάρτησης main



Β. Ασκήσεις3. Αρχείο Εγγραφών

Η δομή σημείο περιέχει τα εξής στοιχεία:

- Αν είναι ορατό (0 ή 1)
- Το χρώμα του (από 0 έως 63)
- Τις συντεταγμένες του στον 2Δ χώρο (int)

Κατασκευάστε ένα πρόγραμμα το οποίο:

- Θα δημιουργεί ένα αρχείο με την εξής γραμμογράφηση:
 - Αρχικά θα υπάρχει το πλήθος των εγγραφών του
 - Έπειτα θα ακολουθούν 10.000 εγγραφές τυχαίων σημείων σε δυαδική μορφή.
- Οι δύο πρώτες μεταβλητές (ορατότητα και χρώμα) θα πρέπει να λαμβάνουν όσο το δυνατόν λιγότερο χώρο (bits)

Επαληθεύστε ότι η εγγραφή έχει γίνει σωστά, διαβάζοντας το αρχείο που κατασκευάσατε.