ΕΠΛ232 – Προγραμματιστικές Τεχνικές και Εργαλεία

Δομές Δεδομένων IV (Διπλά Συνδεδεμένες Λίστες)

Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου

http://www.cs.ucy.ac.cy/courses/EPL232

Dr. Andreas Aristidou a.aristidou@ieee.org

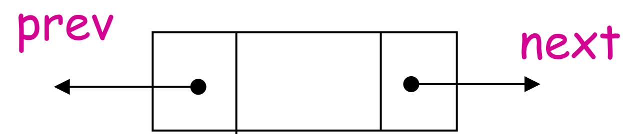
Περιεχόμενο Διάλεξης 15

- **Διπλά Συνδεδεμένες Λίστες**
  - Ορισμοί και Δηλώσεις
  - Υλοποίηση Συνάρτησης `put(l, x, y)` **αναδρομικά** και **επαναληπτικά**
- **Ταξινομημένες Διπλά Συνδεδεμένες Λίστες**
  - Υλοποίηση Συνάρτησης `printlist(l)`
  - Υλοποίηση Συνάρτησης `insert(l, x)`
  - Υλοποίηση Συνάρτησης `delete(l, x)`
- **Κυκλικά Συνδεδεμένες Λίστες**

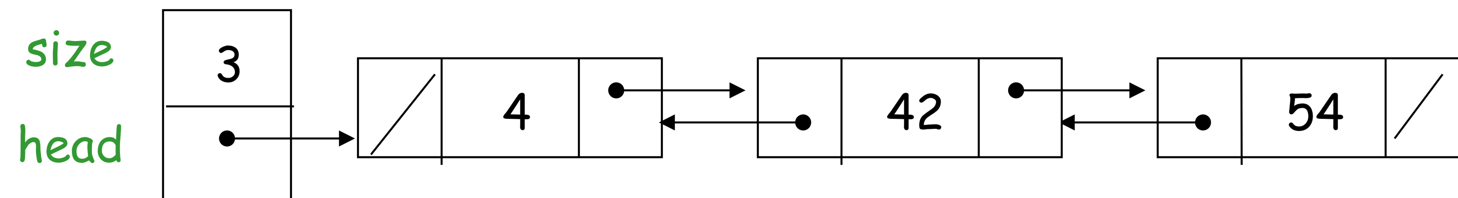
Dr. Andreas Aristidou a.aristidou@ieee.org

Διπλά Συνδεδεμένες Λίστες

- Διπλά Συνδεδεμένη Λίστα (doubly-linked list) ονομάζεται μια λίστα κάθε κόμβος της οποίας κρατά πληροφορίες και για τον επόμενο και για τον προηγούμενο κόμβο:



- Με αυτό τον τρόπο δίνεται η **ευχέρεια μετακίνησης** μέσα στη λίστα και **προς τις δύο κατευθύνσεις**.
- **Παράδειγμα Λίστας:**



Dr. Andreas Aristidou a.aristidou@ieee.org

Διπλά Συνδεδεμένες Λίστες

- Διπλά Συνδεδεμένη Λίστα (doubly-linked list) ονομάζεται μια λίστα κάθε κόμβος της οποίας κρατά πληροφορίες και για τον επόμενο και για τον προηγούμενο κόμβο:

- Η διπλά συνδεδεμένη λίστα καθιστά ευκολότερη τη διαχείριση των κόμβων (εισαγωγή – διαγραφή), γεγονός ιδιαίτερα σημαντικό σε προβλήματα ταξινόμησης και αναζήτησης. Επίσης, τα περιεχόμενα των κόμβων προσπελαύνονται και από τις δύο κατευθύνσεις. Παρουσιάζει το μειονέκτημα της πολυπλοκότητας στη διαχείριση των κόμβων και της επιπρόσθετης απαιτούμενης μνήμης για κάθε κόμβο.

Dr. Andreas Aristidou a.aristidou@ieee.org

Διπλά Συνδεδεμένες Λίστες

- Ποιες δομές χρειάζονται για υλοποίηση μιας διπλά συνδεδεμένης λίστας;
- Ένας κόμβος ορίζεται από το πιο κάτω structure:

```
typedef struct dnode {
        int       data;
        struct dnode *prev;
        struct dnode *next;
        } DNODE;
```

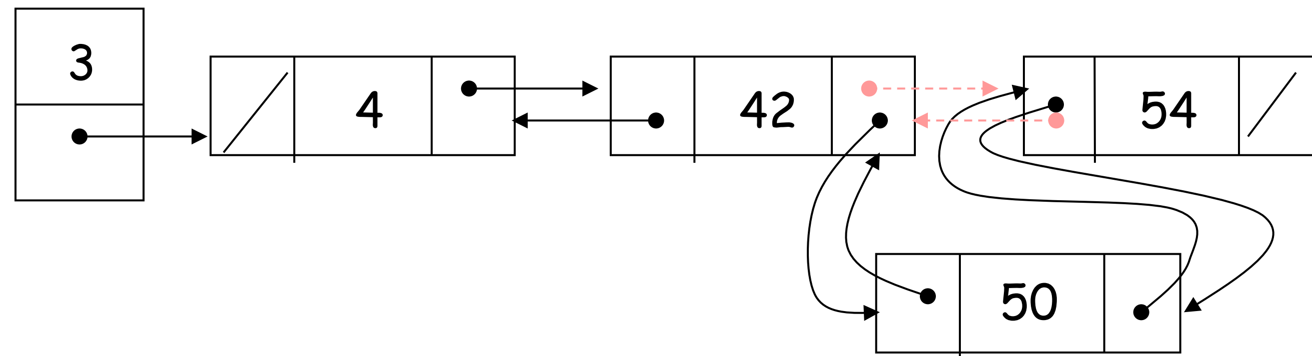
- Ο κόμβος που ορίζει τη **διπλά συνδεδεμένη λίστα** είναι ο ίδιος με αυτό που ορίζει μια στοίβα ή λίστα:

```
typedef struct dllist {
        DNODE *head;
        int   size;
} DLLIST;
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Διπλά Συνδεδεμένες Λίστες

- Προφανώς η **εισαγωγή** στοιχείου σε **κάποιο σημείο** μιας διπλά συνδεδεμένης λίστας περιέχει κάποια **επιπλέον πολυπλοκότητα** από την εισαγωγή σε μια **μονά-συνδεδεμένη** λίστα.
- Αυτό γιατί κάθε **νέος κόμβος** πρέπει να **συνδεθεί** και με τον **επόμενο** και με τον **προηγούμενο** κόμβο στη λίστα. Παρόμοια, κατά τις εξαγωγές στοιχείων.
- **Παράδειγμα εισαγωγής** του στοιχείου **50** μετά το **42** στη λίστα:

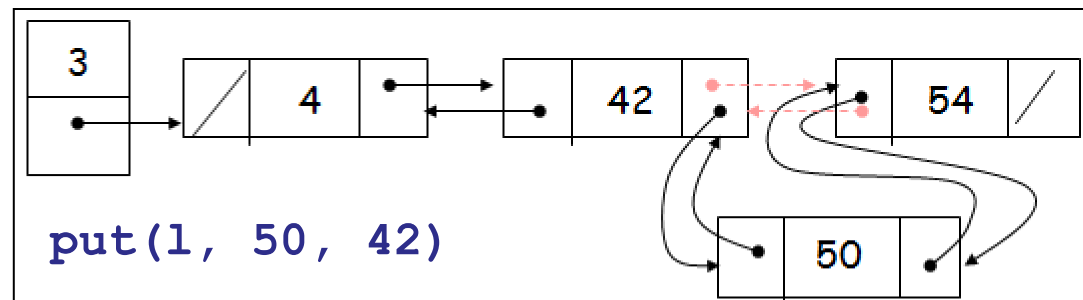


Dr. Andreas Aristidou a.aristidou@ieee.org

Η συνάρτηση put

- Να ορίσετε συνάρτηση **put(l, x, y)** η οποία τοποθετεί το στοιχείο **x** μετά από το στοιχείο **y** μέσα στη λίστα **l**, αν το στοιχείο **y** υπάρχει στην λίστα.

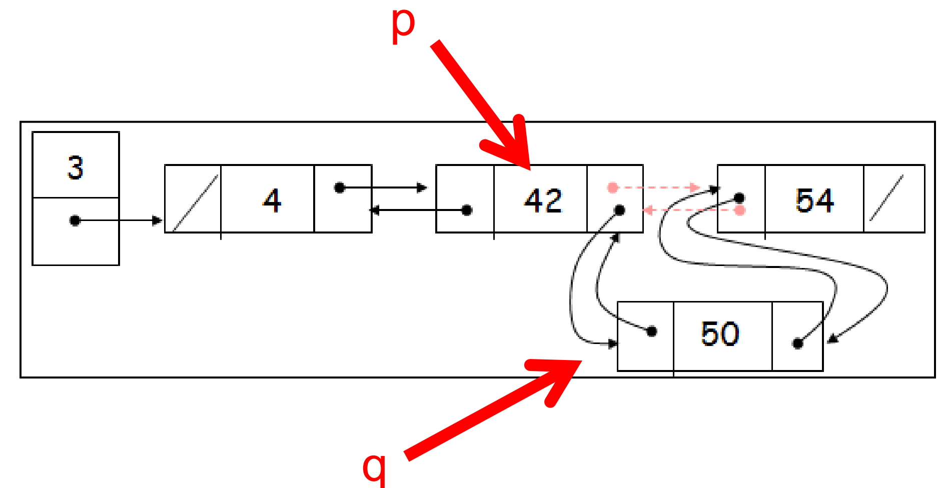
```
void put(DLLIST *l, int x, int y){
  if (l == NULL || l->head == NULL)
     printf("The list is empty, no insertion was made\n");
  else {
     putnode(l->head, x, y, &(l->size));
  }
}
```



Dr. Andreas Aristidou a.aristidou@ieee.org

Η συνάρτηση **puttnode** (Επαναληπτική)

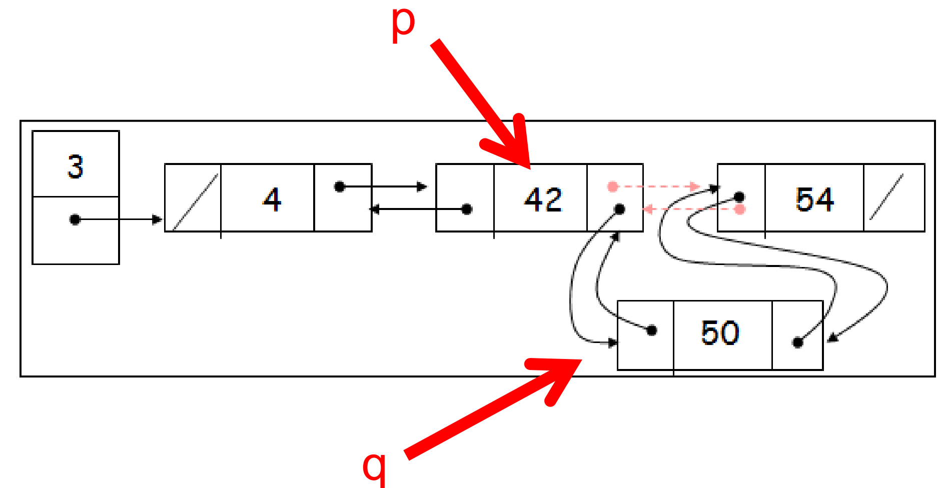
```
void  putnode(DLNODE *p, int x, int y, int *size){
  DLNODE  *q = NULL;
  while((p != NULL) && (p->data != y))
        p = p->next;

  if (p == NULL){
        printf("Element %d does not exist,
                    no insertion was made\n", y);
  }
  else{
        q = (DLNODE *)malloc(sizeof(DLNODE));
        q->data = x;
        q->next = p->next;
        q->prev = p;
        if (p->next != NULL)
                (p->next)->prev = q;
        p->next = q;
        (*size)++;
  }
}
```



Dr. Andreas Aristidou a.aristidou@ieee.org

Η συνάρτηση **putnode** (Αναδρομική)

```
static int putnode(DLNODE *p, int x, int y, int *size){
  DLNODE  *q = NULL;
  if (p == NULL) {
        printf("Element %d does not exist, y); return EXIT_FAILURE;
  }
  else if ( p->data == y ) {
        q = (DLNODE *)malloc(sizeof(DLNODE));
        if (q == NULL) {
            printf("Error: Malloc"); return EXIT_FAILURE;
        }
        q->data = x;
        q->next = p->next;
        q->prev = p;
        if (p->next != NULL)
            (p->next)->prev = q;
        p->next = q;    (*size)++
        return EXIT_SUCCESS;
  }
  else
        return putnode(p->next, x, y, size);
  }
}
```



Dr. Andreas Aristidou a.aristidou@ieee.org

Ταξινομημένες Διπλά Συνδεδεμένες Λίστες

Άσκηση: Να ορίσετε τις **αναδρομικές** διαδικασίες:

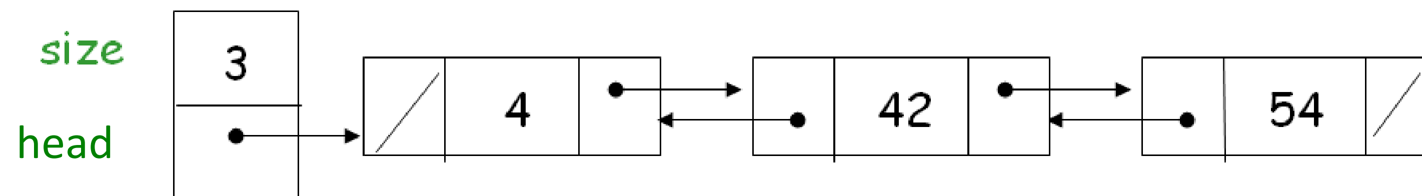
- `void printlist(DLLIST *l)`, η οποία τυπώνει όλα τα στοιχεία της λίστας l.
- `void insert(DLLIST *l, int x)`, η οποία εισάγει το στοιχείο x μέσα στη λίστα l,
- `void delete(DLLIST *l, int x)`, η οποία αφαιρεί το στοιχείο x από τη λίστα l (αν αυτό υπάρχει) και

- Όλες οι πιο πάνω συναρτήσεις πρέπει να διατηρούν *ταξινομημένες* τις διπλά συνδεδεμένες λίστες.

Dr. Andreas Aristidou a.aristidou@ieee.org

Η Συνάρτηση **printList**

```
void printlist(DLLIST *l){
  if (l == NULL || l->size == 0)
      printf("The list is empty\n");
  else
        printnode(l->head);
}
void printnode(DLNODE *p){
  if (p != NULL){
      printf("%d", p->data);
        printnode(p->next);
  }
}
```

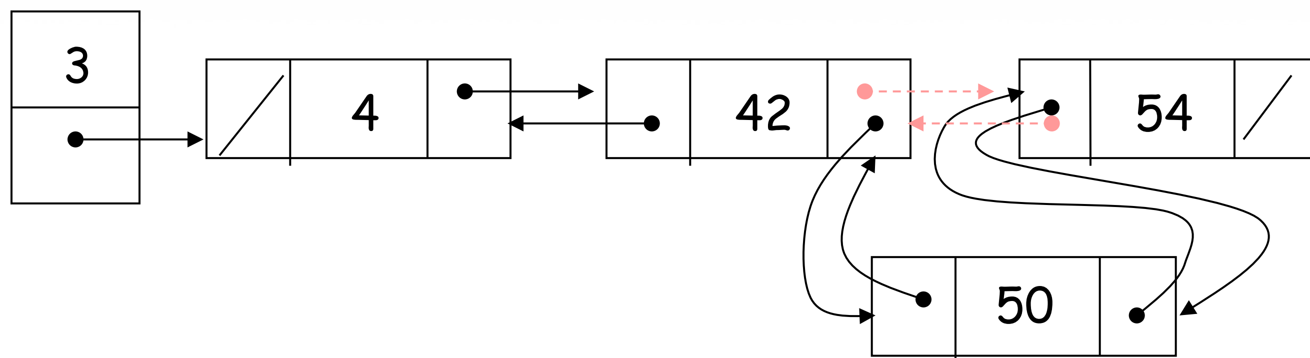


Dr. Andreas Aristidou a.aristidou@ieee.org

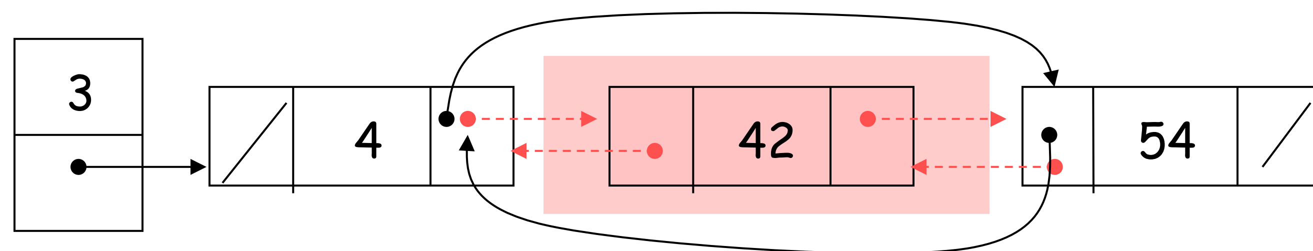
Παραδείγματα

Θα δούμε κάποιους διαφορετικούς τρόπους υλοποίησης. Επιλέξτε τον τρόπο που σας βολεύει καλύτερα.

- **Εισαγωγή** του **50** στη λίστα:

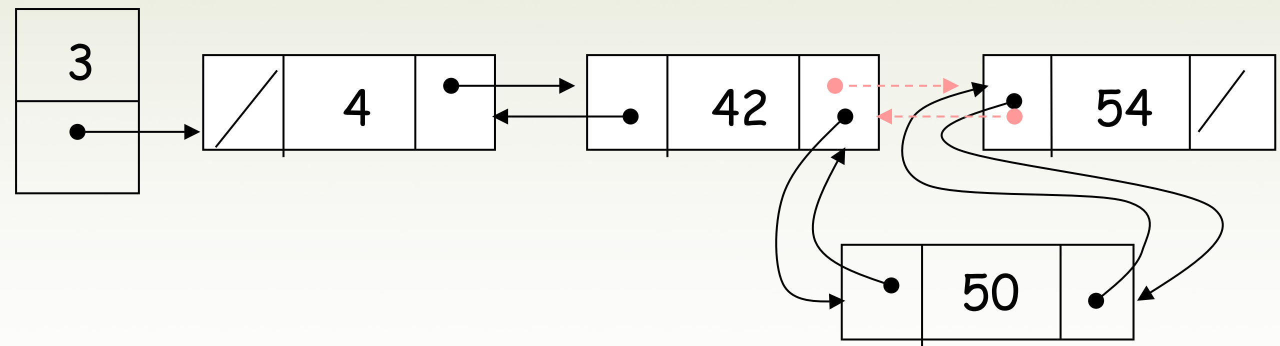


- **Εξαγωγή** του **42** από τη λίστα:



Dr. Andreas Aristidou a.aristidou@ieee.org

Εισαγωγή στοιχείου



- Η εισαγωγή στοιχείου (τον κόμβο τον διαχειρίζεται ο newnode) στην αρχή της λίστας γίνεται ως εξής:

```
head->prev = newnode;
newnode->next = head;
head = newnode;
newnode->prev = NULL;
```

- Αντίστοιχα γίνεται και η εισαγωγή κόμβου στο τέλος της λίστας:

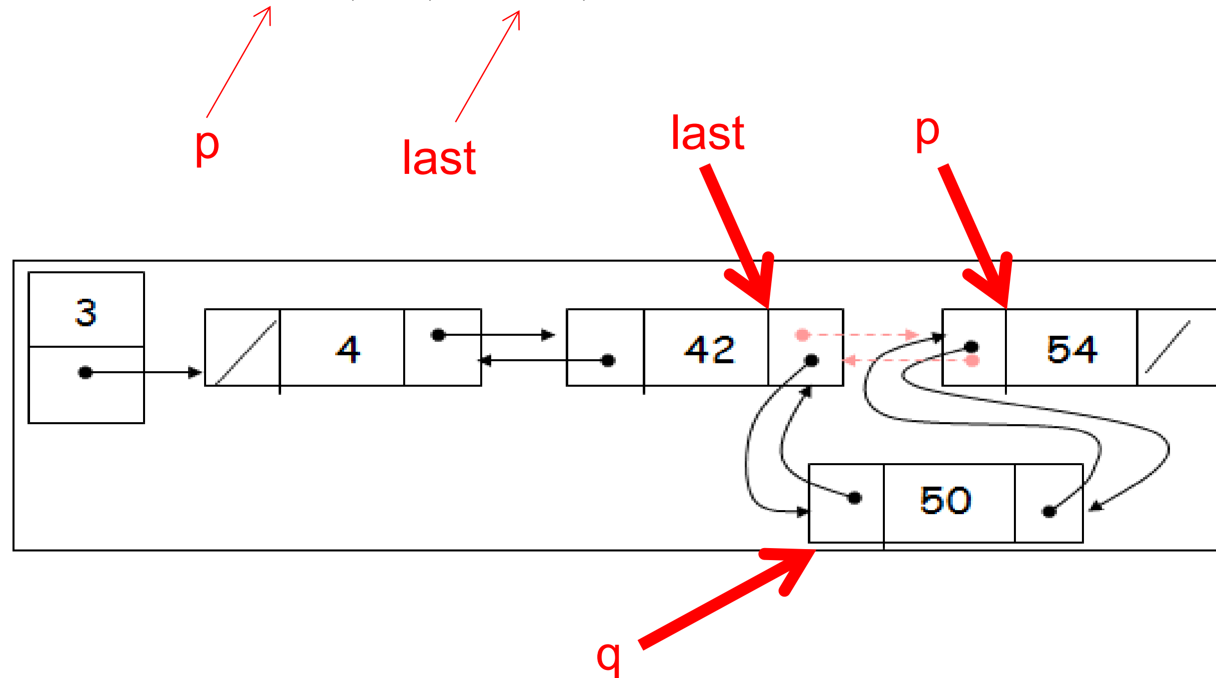
```
tail->next = newnode;
newnode->prev = tail;
tail = newnode;
newnode->next = NULL;
```

- Για την εισαγωγή κόμβου σε ενδιάμεση θέση θεωρείται ότι ο δείκτης current δείχνει στον κόμβο μετά τον οποίο θα γίνει η εισαγωγή, οπότε ο current->next δείχνει στον κόμβο που θα ακολουθεί τον εισαγόμενο:

```
newnode->next = current->next;
(current->next)->prev = newnode;
current->next = newnode;
newnode->prev = current->next;
```

Η Συνάρτηση **insert**

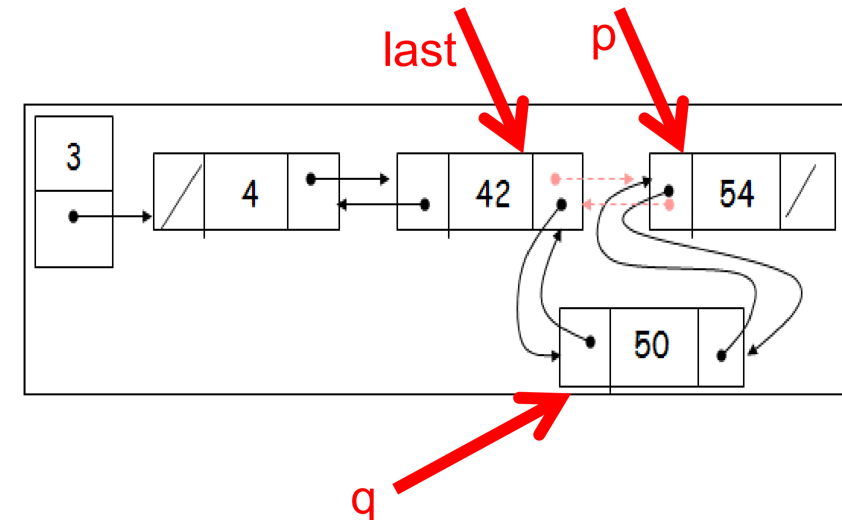
```
void insert(DLLIST *l, int x) {
 if (l == NULL) {
      printf("list is empty\n"); return;
 }
 l->head = insertnode(l->head, x, NULL, &(l->size));
}
```



Dr. Andreas Aristidou a.aristidou@ieee.org

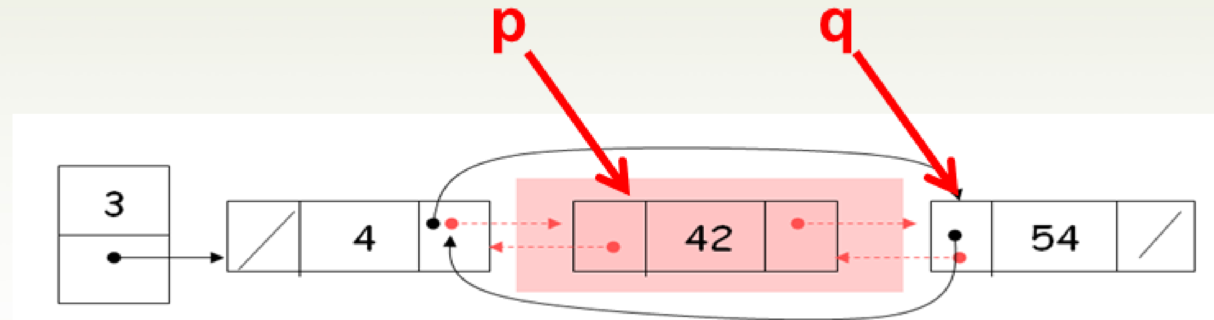
Η Συνάρτηση **insert**

```
DLNODE *insertnode(DLNODE *p,  int x, DLNODE *last, int *size) {
   DLNODE *q = NULL;
   if (p == NULL){// Φτάσαμε στο τέλος της λίστας
        q = (DLNODE *) malloc(sizeof(DLNODE));
        q->data = x; q->next = NULL;
        q->prev = last; (*size)++;
   }
   else if (p->data >= x) { // Περάσαμε το σημείο εισαγωγής
        q = (DLNODE *) malloc (sizeof(DLNODE));
        q->data = x;
        q->next = p;
        q->prev = p->prev;
        if ((p->prev) != NULL)
            (p->prev)->next = q;
        p->prev = q; (*size)++;
   }
   else {
        q = p; // για ομοιόμορφο return στο τέλος
        q->next = insertnode(q->next, x, q, size);
   }
   return q; // για όλους τους κόμβους (εκτός του κόμβου
   εισαγωγής) αυτό είναι το p ουσιαστικά
}
```



Dr. Andreas Aristidou a.aristidou@ieee.org

Διαγραφή στοιχείου



- Η διαγραφή του κόμβου της κεφαλής γίνεται ως εξης:

```
current=head; /* τοποθετείται σε δείκτη, ο οποίος θα επιτελέσει την
απελευθέρωση μνήμης */
head=head->next;
head->prev=NULL;
free(current);
```

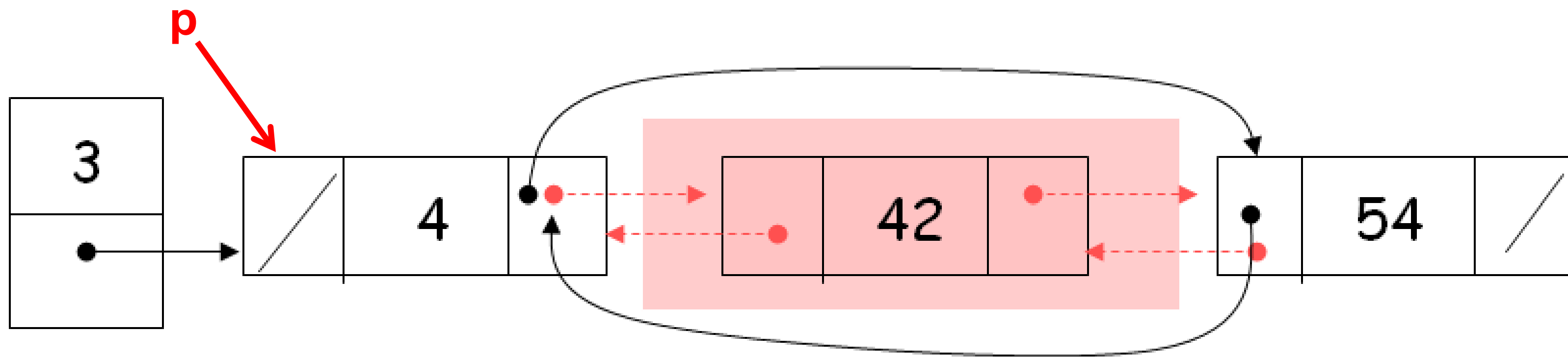
- Κατ’ αντίστοιχο τρόπο διαγράφεται ο τελευταίος κόμβος της λίστας:

```
newnode=tail;
tail=tail->prev;
tail->next=NULL;
free(newnode);
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Η Συνάρτηση **delete**

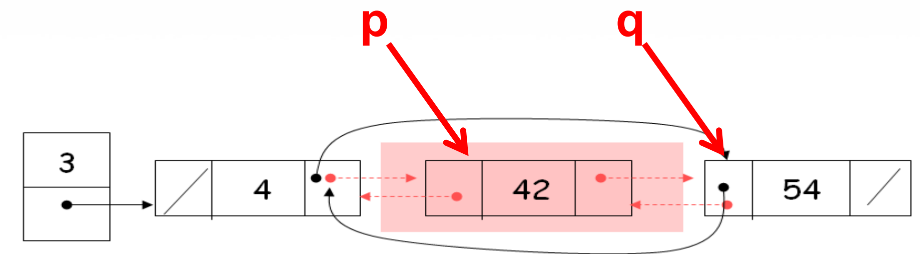
```
void delete(DLLIST *l, int x){
 l->head = deletenode(l->head, x, &(l->size));
}
```



Dr. Andreas Aristidou a.aristidou@ieee.org

Η Συνάρτηση **delete**

```
DLNODE *deletenode(DLNODE *p, int x, int *size){
  DLNODE  *q = NULL;
  if ( (p == NULL) || (p->data > x) ){
        printf("Item %d not found \n", x);
  }
  else if ( p->data == x ){
        q = p->next;
        if (q != NULL)
               q->prev = p->prev;
        if ((p->prev)!= NULL)
              (p->prev)->next = q;
       free(p);
        printf("Item %d has been deleted\n", x);
        (*size)--;
        p = q; // θα γίνει return το p.
  }
  else {
        p->next = deletenode(p->next, x, size);
  }
  return p; // απαιτείται για επιβεβαίωση του backlink σε διαγραφές
}
```

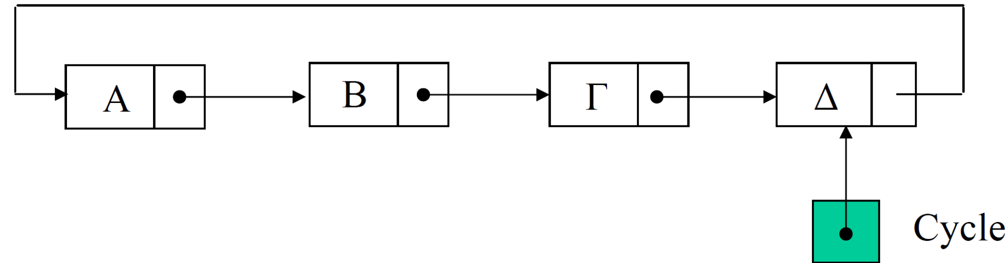


Κυκλικά Απλά Συνδεδεμένες Λίστες

- Η κυκλική λίστα είναι μία απλά συνδεδεμένη λίστα, με τη διαφορά ότι το μέλος next του τελευταίου κόμβου δεν έχει τιμή NULL αλλά τη διεύθυνση του πρώτου κόμβου. Η δομή του κόμβου είναι ίδια με αυτή της απλά συνδεδεμένης λίστας.
- Η διπλά συνδεδεμένη λίστα παρέχει ευελιξία λόγω του σχήματός της και βρίσκει εφαρμογή σε διαδικασίες, όπως ο διαμοιρασμός χρόνου από το λειτουργικό σύστημα.

Κυκλικά Απλά Συνδεδεμένες Λίστες

- Μια κυκλική λίστα μπορεί να παίξει το ρόλο της ουράς, της οποίας και τα δύο άκρα είναι προσιτά με τη βοήθεια ενός μόνο δείκτη.



- Ο κόμβος που ορίζει την κυκλική απλά συνδεδεμένη λίστα είναι:

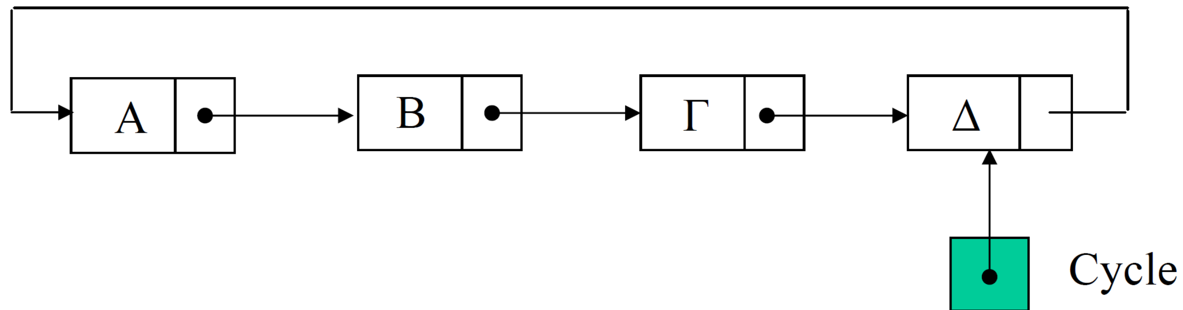
```
typedef struct clist {
DLNODE *back;
…
} CLIST;
```

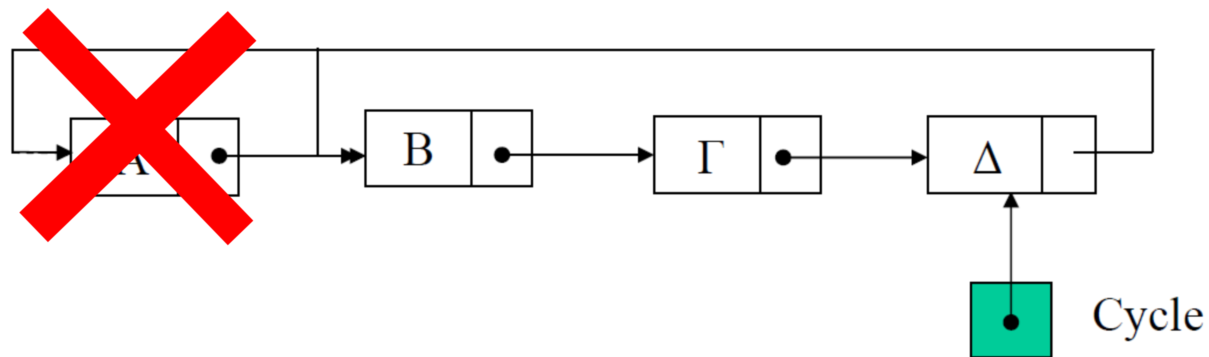
- Το πίσω άκρο, δηλαδή το άκρο όπου γίνονται οι εισαγωγές, είναι αυτό που δείχνεται από τον δείκτη στη λίστα (Cycle).
- Οι εξαγωγές γίνονται ακριβώς μετά από τον κόμβο που δείχνεται από τον δείκτη Cycle.

Dr. Andreas Aristidou a.aristidou@ieee.org

Κυκλικά Απλά Συνδεδεμένες Λίστες

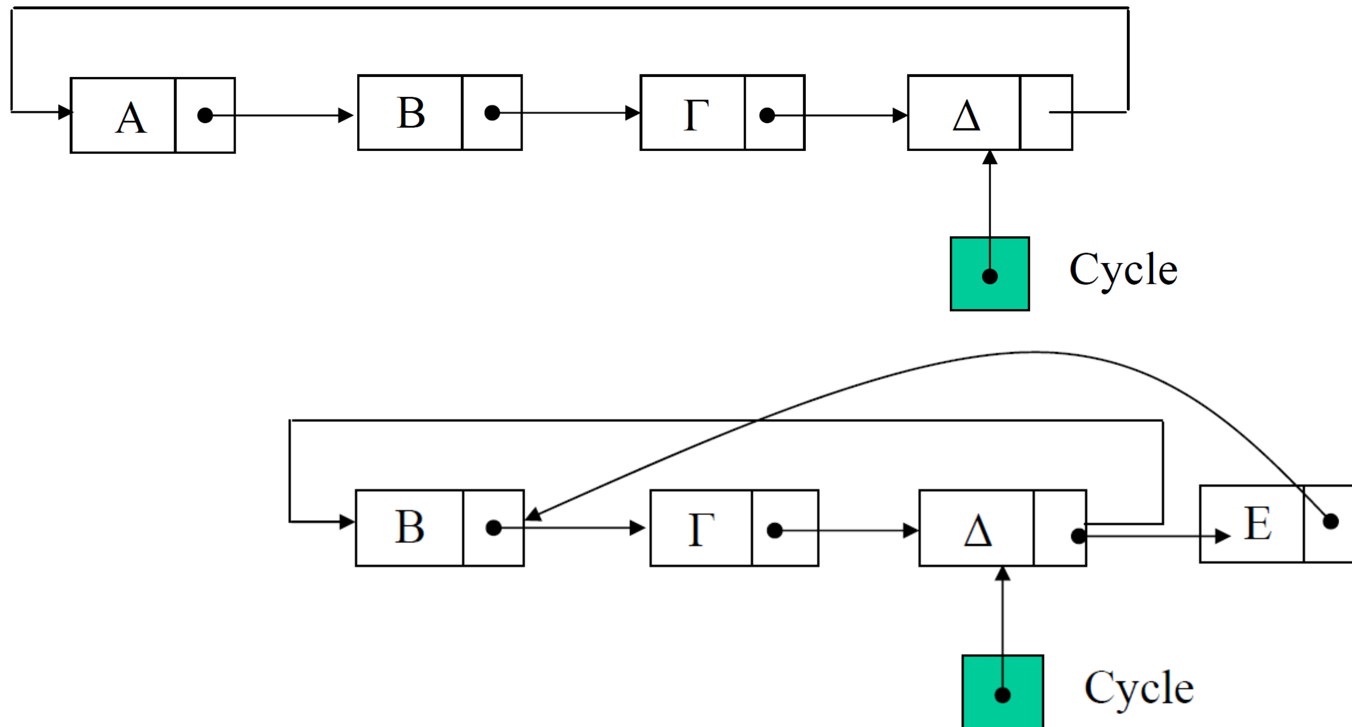
- Διαγραφή/Εξαγωγή Κόμβου





Κυκλικά Απλά Συνδεδεμένες Λίστες

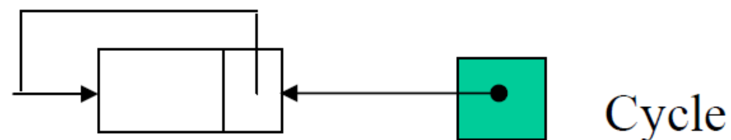
- Εισαγωγή/Προσθήκη Κόμβου



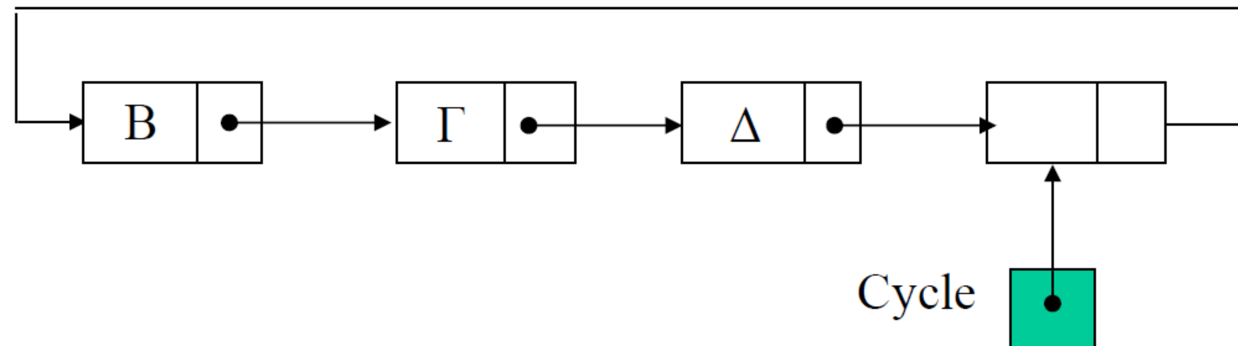
Dr. Andreas Aristidou a.aristidou@ieee.org

Κυκλικά Απλά Συνδεδεμένες Λίστες

- Υπάρχει η εξής ασυνέχεια σε κυκλικά συνδεδεμένες λίστες: ενώ η μη-κενή λίστα δεν έχει καμιά μηδενική τιμή συνδέσμων, η κενή λίστα παριστάνεται από ένα μηδενικό σύνδεσμο. Αυτή η ασυνέχεια προκαλεί πρόσθετους ελέγχους στις διάφορες χρήσιμες πράξεις.
- Για να επιτύχουμε ομοιομορφία μεταξύ των δυο ειδών λίστας μπορούμε να εισαγάγουμε **κόμβους κεφαλής**, δηλαδή κόμβους που δεν περιέχουν πληροφορίες. Για παράδειγμα η άδεια ουρά δίνεται ως:

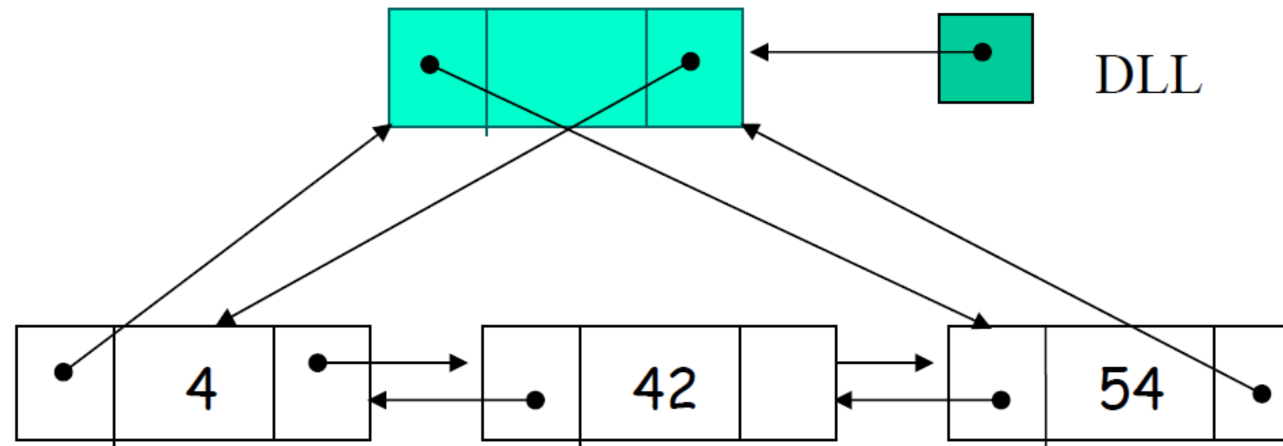


ενώ η ουρά υλοποιείται ως



Κυκλικά Διπλά Συνδεδεμένες Λίστες

- Κυκλική λίστα της οποίας ο κάθε κόμβος δείχνει και στον επόμενο και στον προηγούμενο.
- Με την εισαγωγή κεφαλής σε τέτοιες λίστες παίρνουμε τις κυκλικές διπλά συνδεδεμένες λίστες με κεφαλή.



Dr. Andreas Aristidou a.aristidou@ieee.org

Κυκλικά Διπλά Συνδεδεμένες Λίστες

- Οι διπλά συνδεδεμένες λίστες έχουν το μειονέκτημα πως απαιτούν την ύπαρξη δύο πεδίων δεικτών σε κάθε κόμβο.
- Πολλές από τις αποθηκευμένες πληροφορίες επαναλαμβάνονται: κάθε δείκτης αποθηκεύεται σε δυο κόμβους, τον επόμενο και τον προηγούμενο με αποτέλεσμα τη σπατάλη χώρου.
- Μπορούμε να συμπτύξουμε σε ένα πεδίο τύπου δείκτη τις διευθύνσεις και του επόμενου και του προηγουμένου κόμβου;

→ Τεχνικές Μείωσης Χώρου (Exclusive or)

Dr. Andreas Aristidou a.aristidou@ieee.org