ΕΠΛ232 – Προγραμματιστικές Τεχνικές και Εργαλεία

Δυναμική Δέσμευση Μνήμης και Δομές Δεδομένων (Φροντιστήριο)

Τμήμα Πληροφορικής, Πανεπιστήμιο Κύπρου

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

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 1

- Υποθέστε πως έχουμε μια απλά συνδεδεμένη λίστα με την επόμενη δομή κόμβου:

```
typedef struct node
{
      float weight;
      struct node *next;
}NODE;
```

Ο χειριστής της λίστας NODE δείχνει στην κεφαλή της λίστας. Οι αριθμοί που περιέχει η λίστα είναι τα σωματικά βάρη ενός δείγματος πληθυσμού. Να γραφεί κώδικας ο οποίος να υπολογίζει το μέσο όρο του σωματικού βάρους του δείγματος.

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 1

```
float calculate_mean(NODE *p)
{
        float sum = 0, mean_value;
        int k = 0;
        while(p!=NULL)
        {
                sum = sum + p->weight;
                p = p->next;
                k++;
        }
        mean_value = sum / k;
        return mean_value;
}
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 2

- Τροποποιήστε τη συνάρτηση `delete_from_list` ώστε να χρησιμοποιεί μόνο μία μεταβλητή δείκτη αντί για δύο (`cur` και `prev`).

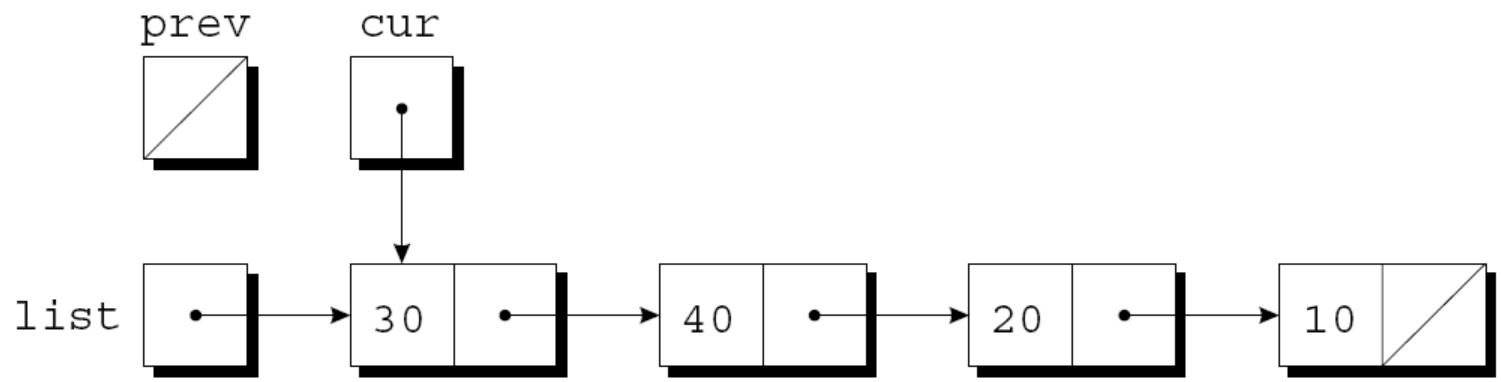
```
struct node *delete_from_list(struct node *list, int n)
{
  struct node *cur, *prev;

  for (cur = list, prev = NULL;
       cur != NULL && cur->value != n;
       prev = cur, cur = cur->next)
    ;
  if (cur == NULL)
    return list;                /* n was not found */
  if (prev == NULL)
    list = list->next;          /* n is in the first node */
  else
    prev->next = cur->next;     /* n is in some other node */
  free(cur);
  return list;
}
```

Dr. Andreas Aristidou a.aristidou@ieee.org

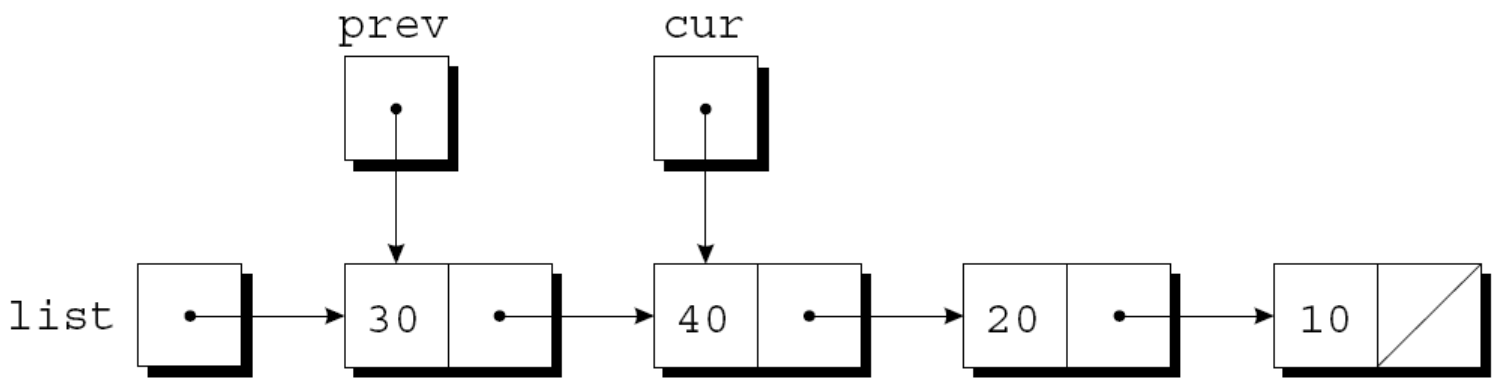
Ασκήσεις κατανόησης 2

- Η συνάρτηση `delete_from_list` χρησιμοποιούσε 2 δείκτες για να έχει πρόσβαση στην προηγούμενη και επόμενη θέση της λίστας, ώστε να μπορεί να παρακάμψει ένα κόμβο.
- Έστω θέλουμε να διαγράψουμε τον κόμβο με τιμή 20.
- Οι `cur = list,` και `prev = NULL` έχουν εκτελεστεί:



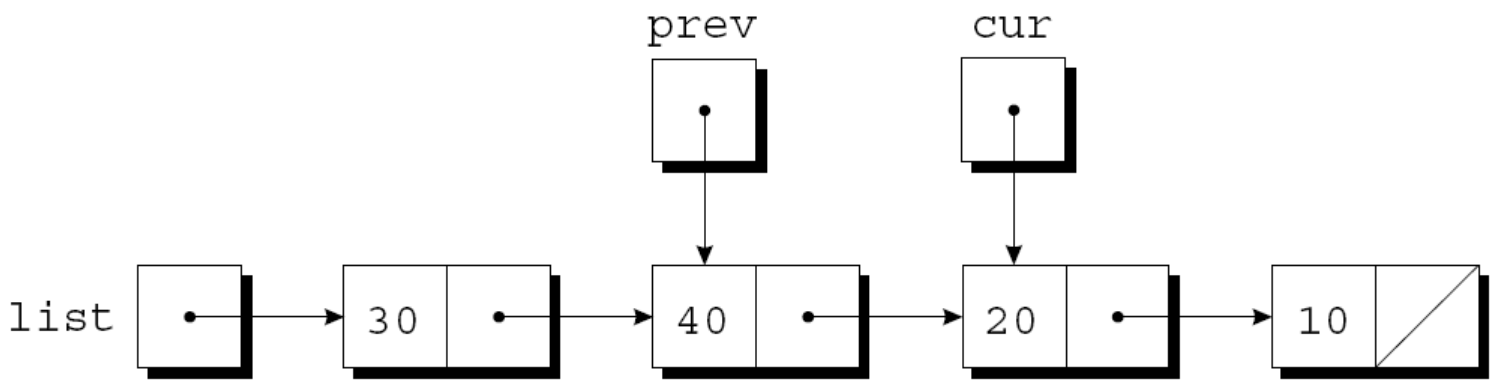
Ασκήσεις κατανόησης 2

- Ο έλεγχος `cur != NULL && cur->value != n` είναι ορθός, αφού η `cur` δείχνει σε έναν κόμβο και ο κόμβος δεν περιέχει το 20.
- Ακολούθως, η `prev = cur, cur = cur->next` έχουν εκτελεστεί:



Ασκήσεις κατανόησης 2

- Ο έλεγχος `cur != NULL && cur->value != n` είναι πάλι σωστός, οπότε `prev = cur, cur = cur->next` εκτελείται ξανά:



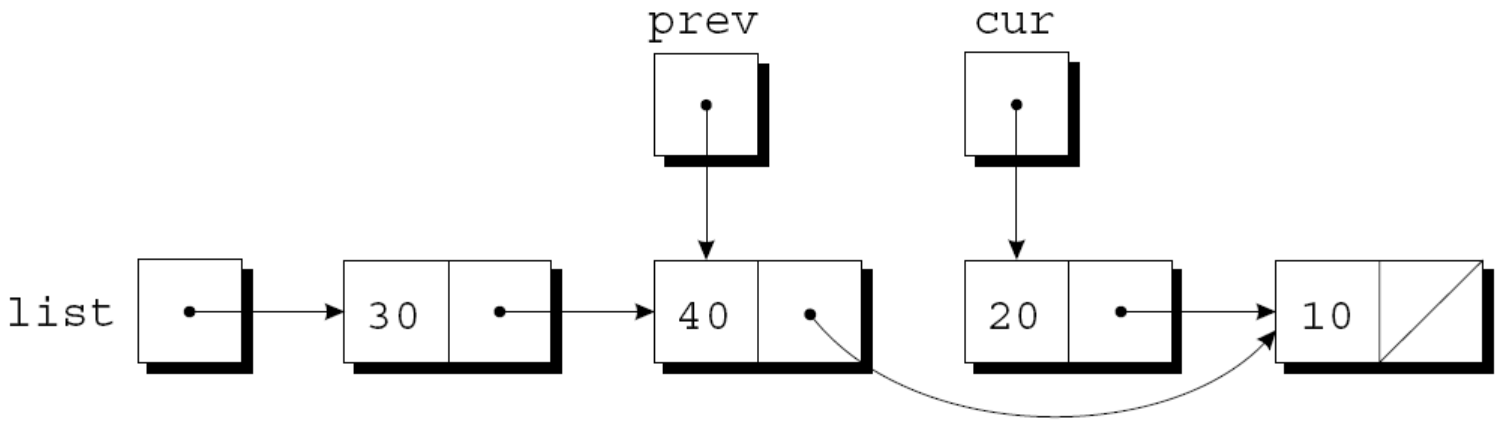
- Αφού το `cur` τώρα δείχνει στον κόμβο που περιέχει το 20, ο έλεγχος `cur->value != n` είναι λάθος και ο βρόγχος τερματίζει.

Ασκήσεις κατανόησης 2

- Στη συνέχεια, θα εκτελέσουμε την παράκαμψη που απαιτείται από το βήμα 2.
- Η δήλωση

```
prev->next = cur->next;
```

κάνει το δείκτη στο προηγούμενο κόμβο να δείχνει στον κόμβο που δίχνει η τρέχουσα τιμή του κόμβου:



Ασκήσεις κατανόησης 2

- Το Βήμα 3 είναι να απελευθερώσει τη μνήμη που καταλαμβάνει ο τρέχων κόμβος:

```
free(cur);
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 2

Η λύση θα είναι αναδρομική:

```
struct node *delete_from_list(struct node *list, int n)
{
  struct node *head;

  if (list != NULL) {
    if (list->value != n)
      list->next = delete_from_list(list->next, n);
    else {
      head = list;
      list = list->next;
      free(head);
    }
  }
  return list;
}
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 3

- Η ακόλουθη συνάρτηση υποτίθεται εισάγει ένα νέο κόμβο στην κατάλληλη θέση μιας ταξινομημένης λίστας, επιστρέφοντας ένα δείκτη στον πρώτο κόμβο της τροποποιημένης λίστας. Ωστόσο, η συνάρτηση δεν δουλεύει σωστά για όλες τις κλάσεις. Εξηγήστε γιατί.

```
typedef struct node
{
        int value;
        struct node *next;
}NODE;
```

```
struct node
*insert_into_ordered_list(struct node
*list, struct node *new_node)
{
  struct node *cur = list, *prev = NULL;

  while (cur->value <= new_node->value)
  {
    prev = cur;
    cur = cur->next;
  }
    prev->next = new_node;
    new_node->next = cur;
    return list;
  }
}
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 3

- Η ακόλουθη συνάρτηση υποτίθεται εισάγει ένα νέο κόμβο στην κατάλληλη θέση μιας ταξινομημένης λίστας, επιστρέφοντας ένα δείκτη στον πρώτο κόμβο της τροπ
συνά
κλάσ

Η συνάρτηση δεν μπορεί να εισαγάγει με επιτυχία κόμβους σε μια κενή λίστα ή στο τέλος μιας λίστας (επειδή δεν ελέγχει αν έχει φτάσει στο τέλος).

```
typedef struct node
{
        int value;
        struct node *next;
}NODE;
```

```
struct node
*insert_into_ordered_list(struct node
*list, struct node *new_node)
{
  struct node *cur = list, *prev = NULL;

                                  ode->value)

    cur = cur->next;
  }
    prev->next = new_node;
    new_node->next = cur;
    return list;
  }
}
```


Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 3

```
struct node *insert_into_ordered_list(struct node *list, struct node
*new_node)
{
  struct node *cur = list, *prev = NULL;

  while (cur != NULL && cur->value <= new_node->value) {
    prev = cur;
    cur = cur->next;
  }
  if (prev == NULL) {      /* insert new node at beginning of list */
    new_node->next = list;
    return new_node;
  } else {                 /* insert new node in middle or at end of list */
    prev->next = new_node;
    new_node->next = cur;
    return list;
  }
}
```

Ασκήσεις κατανόησης 4

- Γράψτε ένα πρόγραμμα σε C για να δημιουργήσετε μια απλά συνδεδεμένη λίστα με `n` κόμβους και να την εμφανίσετε με αντίστροφη σειρά.

Test Data :

```
Input the number of nodes : 3
Input data for node 1 : 5
Input data for node 2 : 6
Input data for node 3 : 7
```

Data entered in the list are :

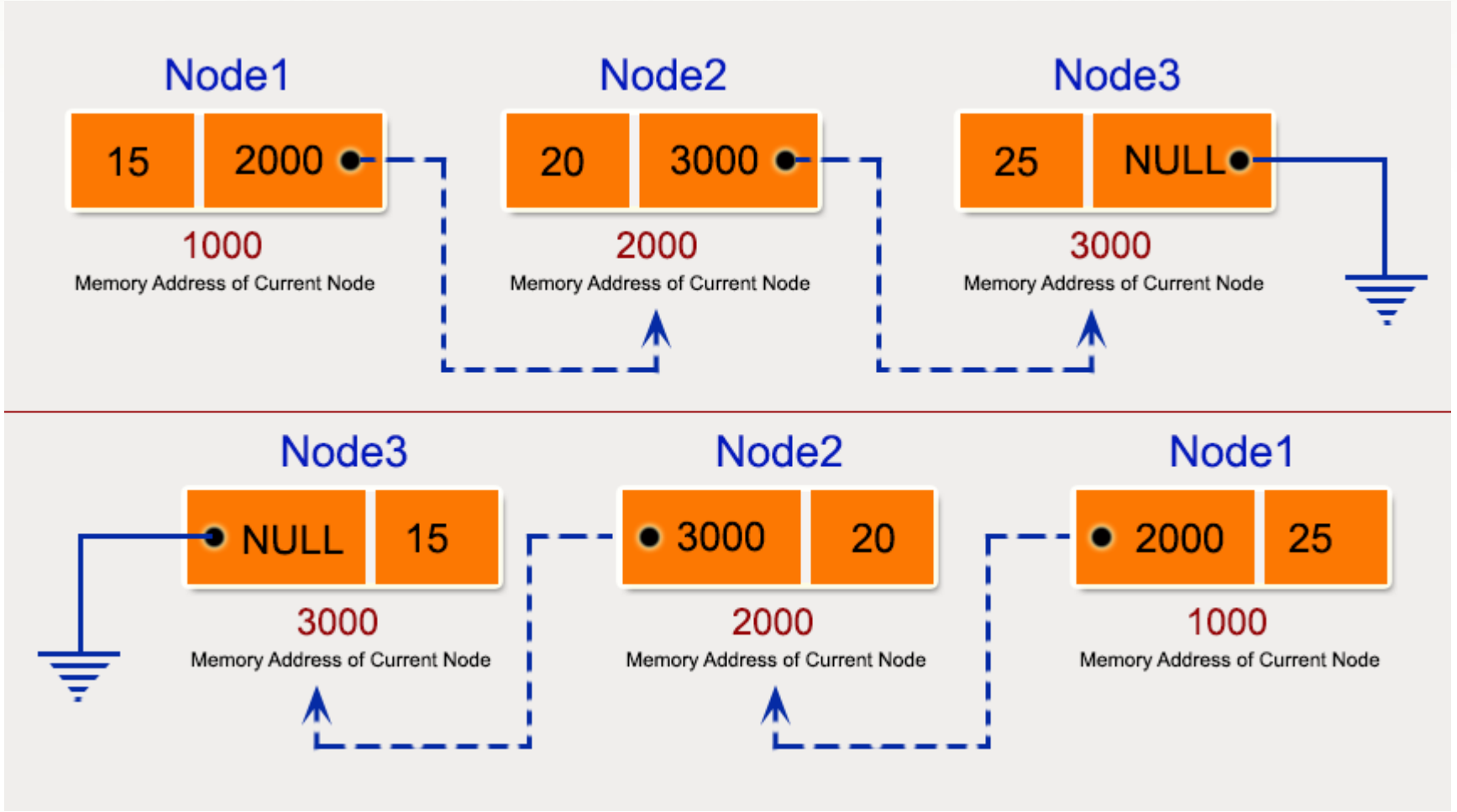
```
Data = 5
Data = 6
Data = 7
```

The list in reverse are :

```
Data = 7
Data = 6
Data = 5
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 4



Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 4

```
#include <stdio.h>
#include <stdlib.h>

typedef struct node
{
    int         num;              //Data of the node
    struct node *nextptr;         //Address of the node
}NODE;

NODE *stnode;

void createNodeList(int n);       //function to create the list
void reverseDispList();           //function to convert the list in reverse
void displayList();               //function to display the list
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 4

```
void createNodeList(int n)
{
    NODE *fnNode, *tmp;
    int num, i;
    stnode = (NODE *)malloc(sizeof(NODE));
    if(stnode == NULL) //check whether the stnode is NULL
    {
        printf(" Memory can not be allocated.");
    }
    else
    {
// reads data for the node through keyboard
        printf(" Input data for node 1 : ");
        scanf("%d", &num);
        stnode-> num = num;
        stnode-> nextptr = NULL; //Links the address field to NULL
        tmp = stnode;
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 4

```
//Creates n nodes and adds to linked list
        for(i=2; i<=n; i++)
        {
            fnNode = (NODE *)malloc(sizeof(NODE));
            if(fnNode == NULL) //check whether the fnnode is NULL
            {
                printf(" Memory can not be allocated.");
                break;
            }
            else
            {
                printf(" Input data for node %d : ", i);
                scanf(" %d", &num);
                fnNode->num = num;       // links the num field of fnNode with num
                fnNode->nextptr = NULL; // links the address field of fnNode with NULL
                tmp->nextptr = fnNode; // links previous node i.e. tmp to the fnNode
                tmp = tmp->nextptr;
            }
        }
    }
}
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 4

```
void reverseDispList()
{
    NODE *prevNode, *curNode;

    if(stnode != NULL)
    {
        prevNode = stnode;
        curNode = stnode->nextptr;
        stnode = stnode->nextptr;

        prevNode->nextptr = NULL; //convert the first node as last

        while(stnode != NULL)
        {
            stnode = stnode->nextptr;
            curNode->nextptr = prevNode;

            prevNode = curNode;
            curNode = stnode;
        }
        stnode = prevNode; //convert the last node as head
    }
}
```

Ασκήσεις κατανόησης 4

```
void displayList()
{
    NODE *tmp;
    if(stnode == NULL)
    {
        printf(" No data found in the list.");
    }
    else
    {
        tmp = stnode;
        while(tmp != NULL)
        {
            printf(" Data = %d\n", tmp->num);    // prints the data of current node
            tmp = tmp->nextptr;                  // advances the position of current node
        }
    }
}
```

Θυμηθείτε την αναδρομική λύση!!

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 4

```
int main()
{
    int n;
    printf("\n\n Linked List : \n");
    printf("------------------ \n");

    printf(" Input the number of nodes : ");
    scanf("%d", &n);
    createNodeList(n);
    printf("\n Data entered in the list are : \n");
    displayList();
    reverseDispList();
    printf("\n The list in reverse are :  \n");
    displayList();
    return 0;
}
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 4

- Γράψτε ένα πρόγραμμα σε C για να δημιουργήσετε μια διπλά συνδεδεμένη λίστα με `n` κόμβους και να την εμφανίσετε με αντίστροφη σειρά.

Test Data :

```
Input the number of nodes : 3
Input data for node 1 : 5
Input data for node 2 : 6
Input data for node 3 : 7
```

Data entered in the list are :

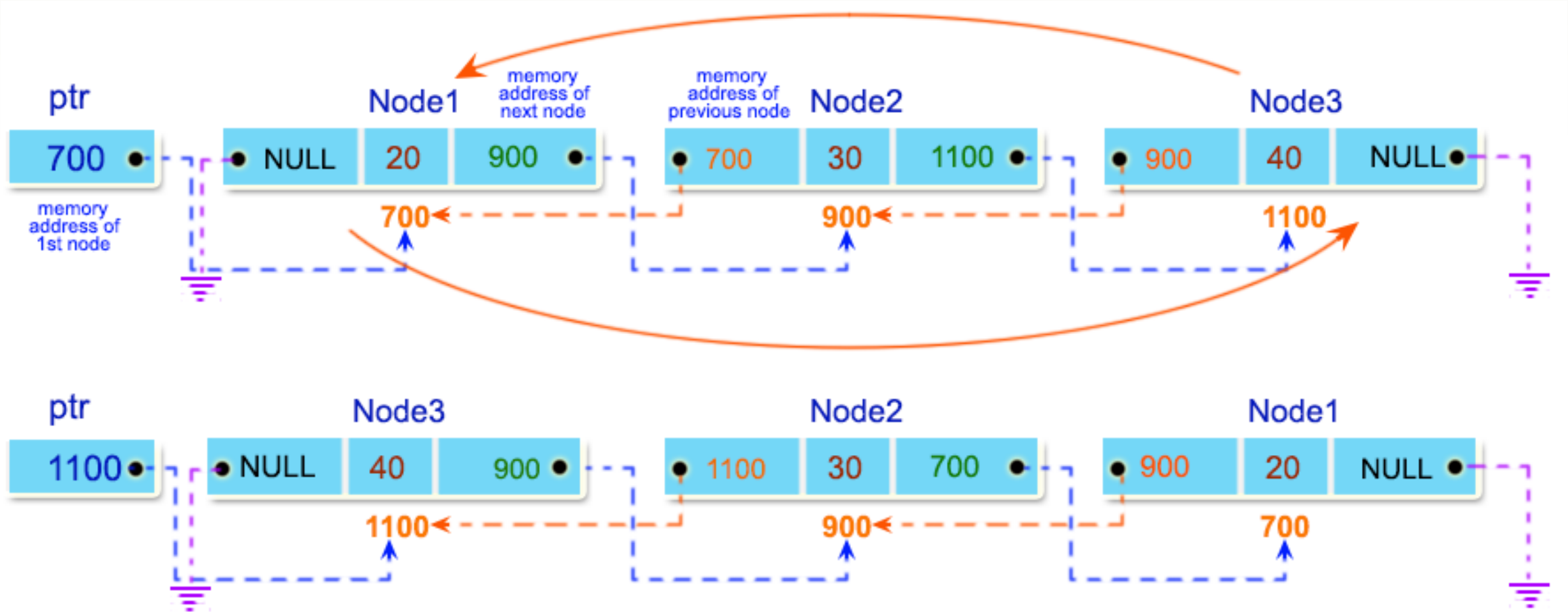
```
Data = 5
Data = 6
Data = 7
```

The list in reverse are :

```
Data = 7
Data = 6
Data = 5
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 4



Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 5

```
#include <stdio.h>
#include <stdlib.h>

typedef struct node {
    int num;
    struct node * preptr;
    struct node * nextptr;
}NODE;

NODE *stnode, *ennode;


void DlListcreation(int n);
void displayDlListRev();
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 5

```
void DlListcreation(int n)
{
    int i, num;
    NODE *fnNode;

    if(n >= 1)
    {
        stnode = (NODE *)malloc(sizeof(NODE));

        if(stnode != NULL)
        {
            printf(" Input data for node 1 : "); // assigning data in the first node
            scanf("%d", &num);

            stnode->num = num;
            stnode->preptr = NULL;
            stnode->nextptr = NULL;
            ennode = stnode;
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 5

```
// putting data for rest of the nodes
            for(i=2; i<=n; i++){
                fnNode = (NODE *)malloc(sizeof(NODE));
                if(fnNode != NULL){
                    printf(" Input data for node %d : ", i);
                    scanf("%d", &num);
                    fnNode->num = num;
                    fnNode->preptr = ennode; // new node is linking with the previous node
                    fnNode->nextptr = NULL;

                    ennode->nextptr = fnNode;// previous node is linking with the new node
                    ennode = fnNode;         // assign new node as last node
                }
                else{
                     printf(" Memory can not be allocated.");
                     break;
                 }
            }
        }
        else{
            printf(" Memory can not be allocated.");
        }
    }
}
```

Ασκήσεις κατανόησης 5

```
void displayDlListRev()
{
    NODE * tmp;
    int n = 0;

    if(ennode == NULL)
    {
        printf(" No data found in the List yet.");
    }
    else
    {
        tmp = ennode;
        printf("\n Data in reverse order are :\n");
        while(tmp != NULL)
        {
            printf(" Data in node %d : %d\n", n+1, tmp->num);
            n++;
            tmp = tmp->preptr; // current pointer set with previous node
        }
    }
}
```

Dr. Andreas Aristidou a.aristidou@ieee.org

Ασκήσεις κατανόησης 5

```
int main()
{
    int n;
    stnode = NULL;
    ennode = NULL;
    printf("\n\n Doubly Linked List \n");
    printf("---------------------- \n");

    printf(" Input the number of nodes : ");
    scanf("%d", &n);

    DlListcreation(n);
    displayDlListRev();
    return 0;
}
```

Dr. Andreas Aristidou a.aristidou@ieee.org