ΤΕΙ ΚΕΝΤΡΙΚΗΣ ΜΑΚΕΔΟΝΙΑΣ

ΑΝΟΙΚΤΑ ΑΚΑΔΗΜΑΙΚΑ ΜΑΘΗΜΑΤΑ

Αλγόριθμοι και Δομές Δεδομένων**(Θ)**

Ενότητα 9: **ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ**

Ευάγγελος Γ. Ούτσιος

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ
ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΤΕ

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο

ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ & ΘΡΗΣΚΕΥΜΑΤΩΝ, ΠΟΛΙΤΙΣΜΟΥ & ΑΘΛΗΤΙΣΜΟΥ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ

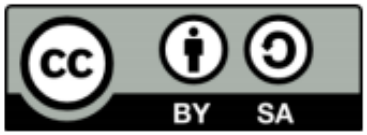
ΕΣΠΑ 2007-2013
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ

Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης

Άδειες Χρήσης

- Το παρόν εκπαιδευτικό υλικό υπόκειται σε άδειες χρήσης Creative Commons.

- Για εκπαιδευτικό υλικό, όπως εικόνες, που υπόκειται σε άλλου τύπου άδειας χρήσης, η άδεια χρήσης αναφέρεται ρητώς.



Χρηματοδότηση

- Το παρόν εκπαιδευτικό υλικό έχει αναπτυχθεί στα πλαίσια του εκπαιδευτικού έργου του διδάσκοντα.
- Το έργο «Ανοικτά Ακαδημαϊκά Μαθήματα στο ΤΕΙ Κεντρικής Μακεδονίας» έχει χρηματοδοτήσει μόνο τη αναδιαμόρφωση του εκπαιδευτικού υλικού.
- Το έργο υλοποιείται στο πλαίσιο του Επιχειρησιακού Προγράμματος «Εκπαίδευση και Δια Βίου Μάθηση» και συγχρηματοδοτείται από την Ευρωπαϊκή Ένωση (Ευρωπαϊκό Κοινωνικό Ταμείο) και από εθνικούς πόρους.

ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ

Οι στατικές δομές που μελετήθηκαν, παρουσιάζουν προβλήματα στην εισαγωγή και διαγραφή κόμβων και στην αποδοτική εκμετάλλευση της διαθέσιμης μνήμης.

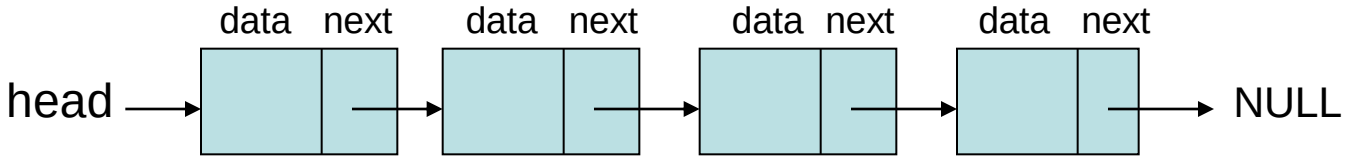
Κύριο χαρακτηριστικό των συνδεδεμένων λιστών είναι ότι οι κόμβοι τους βρίσκονται σε απομακρυσμένες θέσεις μνήμης και η σύνδεσή τους γίνεται με δείκτες.

ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ

Κάθε κόμβος της λίστας υλοποιείται με μία δομή (structure) με δύο στοιχεία.

Το ένα μέλος (data) περιέχει τα δεδομένα (οποιουδήποτε τύπου) του κόμβου και το άλλο μέλος (next) είναι δείκτης προς τον επόμενο κόμβο.

Τοποθετείται ένας δείκτης (head) στον πρώτο κόμβο για να προσπελάζεται η λίστα, ενώ ο δείκτης του τελευταίου κόμβου δείχνει στο NULL για να εντοπίζεται το τέλος της λίστας.



ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ

Παράδειγμα υλοποίησης λίστας ακεραίων στην C:

```
struct node
{
   int  data;
   struct node *next;
};
typedef struct node * PTR;
```

ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ

Δημιουργία λίστας

```
PTR  list_create(PTR  head)
{
  PTR   current;
  int  x;
  printf("Give an integer, 0 to stop:");
  scanf("%i",&x);
  if (x = = 0)
     return  NULL;
  else
    {
      head = malloc(sizeof(struct node));
      head->data = x;
      current = head;
      printf("Give an integer, 0 to stop:");
      scanf("%i",&x);
while (x!=0)
        {
          current->next = malloc(sizeof(struct
      node));
          current = current->next;
          current->data = x;
          printf("Give an integer, 0 to stop:");
             scanf("%i",&x);
           }
       current->next = NULL;
     }
  return head;
}
```

ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ

Εισαγωγή στοιχείου

```
PTR insert_to_list(PTR head, int  x)
{
  PTR current, previous, newnode;
  int found;
  newnode = malloc(sizeof(struct
    node));
  newnode->data = x;
  newnode->next = NULL;
  if (head = = NULL)
      head = newnode;
  else
     if (newnode->data < head->data)
      {
         newnode->next = head;
         head = newnode;
      }
 else
{
        previous = head;
        current = head->next;
        found = 0;
        while (current != NULL && found = = 0)
        {
           if (newnode->data < current->data)
              found = 1;
           else
             {
               previous = current;
               current = current->next;
             }
         }
        previous->next = newnode;
        newnode->next = current;
     }
  return head;
}
```

ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ

Διαγραφή στοιχείου από λίστα

```
PTR  delete_from_list(PTR head, int  x)
{
  PTR current, previous;
  int found;
  current = head;
  if (current = = NULL)
    {
      printf("Empty list...nothing to
     delete.\n");
      getch();
    }
  else
      if (x = = head->data)
      {
          head = head->next;
          free(current);
      }
      else
         {
           previous = current;
           current = head->next;
           found = 0;
           while (current != NULL && found = = 0)
              {
                if (x = = current->data)
                    found = 1;
                else
                    {
                     previous = current;
                     current = current->next;
                    }
              }
            if  (found = = 1)
                {
                   previous->next = current->next;
                   free(current);
                }
                else
                {
                   printf("\nThe character is not in the
           list.");
                   getch();
                }
             }
    return head;
}
```

ΣΥΝΔΕΔΕΜΕΝΕΣ ΛΙΣΤΕΣ

Εκτύπωση λίστας

```
void  print_list(PTR head)
{
   PTR current;
   current = head;
   if (current = = NULL)
        printf("The list is empty.\n");
   else
        while (current != NULL)
        {
          printf("%i ", current->data);
          current = current->next;
        }
}
```