Δομές Δεδομένων

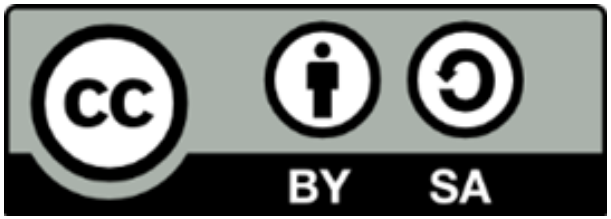
Ενότητα 6: Εφαρμογή Συνδεδεμένων Λιστών: Αλφαβητικό ευρετήριο κειμένου- Υλοποίηση ΑΤΔ Στοίβα και Ουρά με δείκτες

Καθηγήτρια Μαρία Σατρατζέμη

Τμήμα Εφαρμοσμένης Πληροφορικής

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 Πανεπιστήμιο 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.

Σκοποί ενότητας

- Να χρησιμοποιήσουν Συνδεδεμένες λίστες για να υλοποιήσουν αποτελεσματικά ένα αλφαβητικό ευρετήριο
- Να κατανοήσουν ότι η στοίβα και η ουρά μπορούν να υλοποιηθούν ως συνδεδεμένες δομές

Περιεχόμενα ενότητας

- Αλφαβητικό ευρετήριο κειμένου
- Παράδειγμα αλφαβητικού ευρετηρίου κειμένου
- Αλγόριθμος κατασκευής αλφαβητικού ευρετηρίου
- Υλοποίηση στοίβας και ουράς με δείκτες
- Υλοποίηση στοίβας με δείκτες
- Πακέτο για τον ΑΤΔ Συνδεδεμένη Στοίβα
- Πακέτο για τον ΑΤΔ Συνδεδεμένη Ουρά

Εφαρμογή Συνδεδεμένων Λιστών: Αλφαβητικό ευρετήριο κειμένου

Αλφαβητικό ευρετήριο κειμένου -1-

- Ένα αλφαβητικό ευρετήριο κειμένου (text concordance) είναι μια αλφαβητική λίστα όλων των διαφορετικών λέξεων από ένα κομμάτι κειμένου.
- Το γεγονός ότι οι λέξεις σε ένα αλφαβητικό ευρετήριο είναι ταξινομημένες με αλφαβητική σειρά σημαίνει ότι ένα αλφαβητικό ευρετήριο είναι μια ταξινομημένη λίστα.

Αλφαβητικό ευρετήριο κειμένου -2-

Κατασκευή ευρετηρίου:

- Ξεκινάμε με μια άδεια λίστα.
- Κάθε λέξη διαβάζεται και τοποθετείται στη σωστή θέση μέσα στη λίστα, εφόσον δεν υπάρχει ήδη στη λίστα.
- Προφανώς, μπορεί να χρειαστεί η εισαγωγή λέξεων να γίνει σε οποιοδήποτε σημείο της λίστας, οπότε χρειαζόμαστε μια συνδεδεμένη λίστα.
- Για κάθε λέξη που διαβάζεται πρέπει να γίνει σειριακή αναζήτηση της συνδεδεμένης λίστας ξεκινώντας από τον πρώτο κόμβο.

Αλφαβητικό ευρετήριο κειμένου -3-

- Αν πρόκειται για μεγάλο κείμενο, το ευρετήριο μπορεί να γίνει τόσο μεγάλο που η αναζήτηση μιας και μόνο λίστας δεν είναι αποδοτική.

- Για να μειωθεί ο χρόνος αναζήτησης, μπορούμε να χρησιμοποιήσουμε πολλές μικρές συνδεδεμένες λίστες.

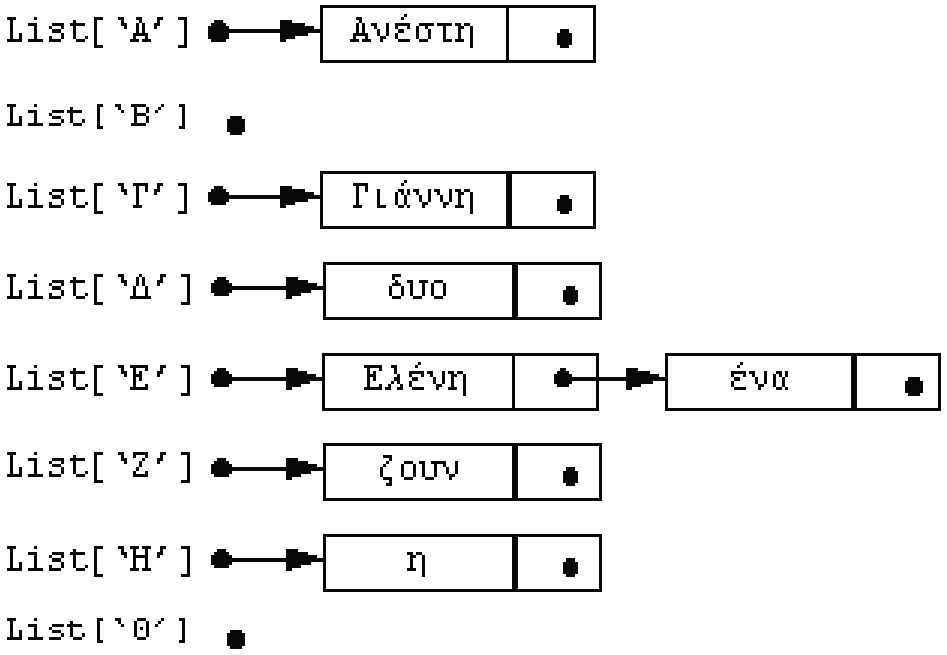
Αλφαβητικό ευρετήριο κειμένου -4-

Για το συγκεκριμένο πρόβλημα:

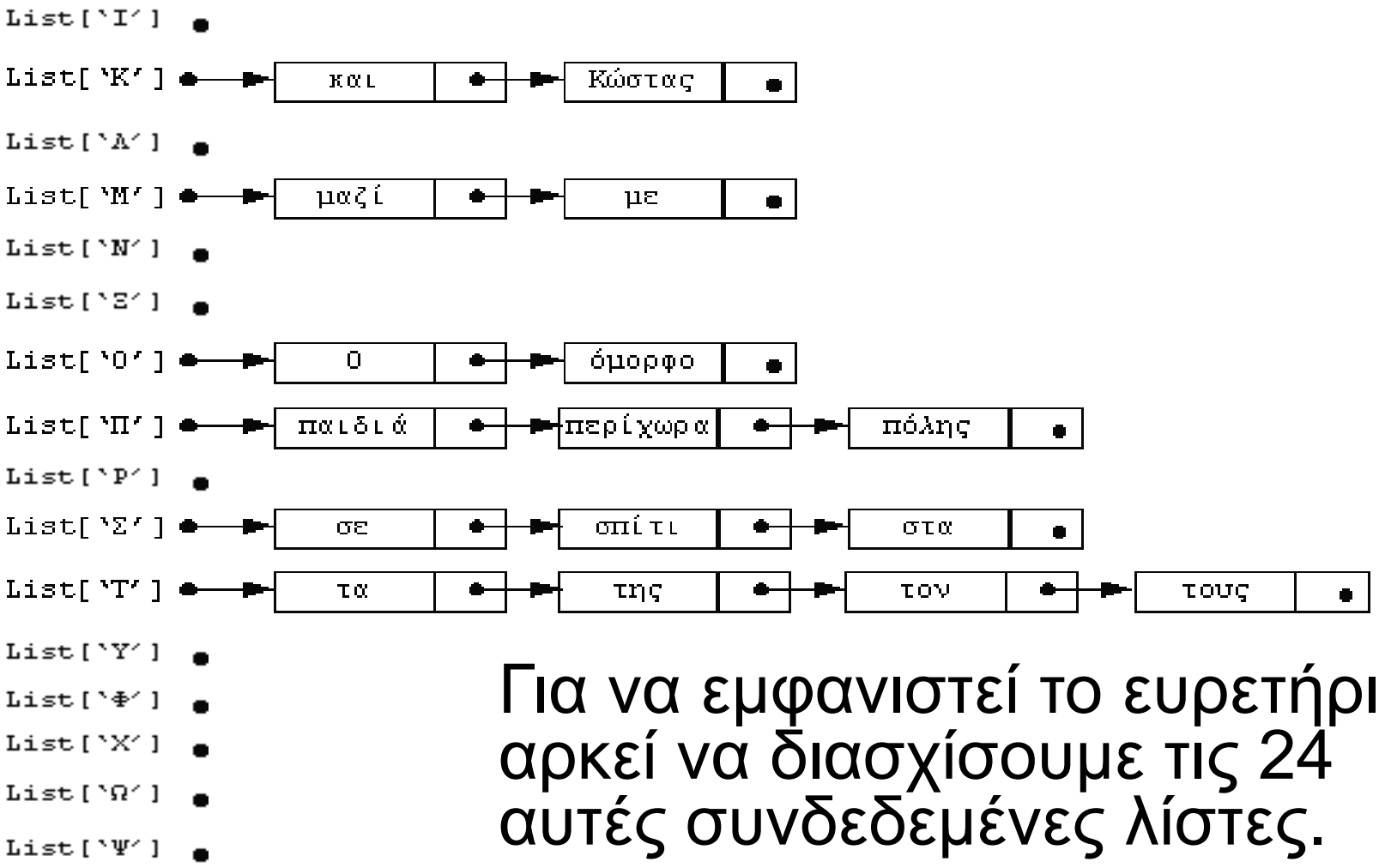
- μπορούμε να κατασκευάσουμε μια συνδεδεμένη λίστα για κάθε γράμμα της αλφαβήτου, δηλαδή μια λίστα με λέξεις που ξεκινάν από "Α", μια λίστα με λέξεις που ξεκινάν από "Β", κ.ο.κ.
- Επομένως, συνολικά θα χρειαστούν 24 συνδεδεμένες λίστες καθώς και 24 δείκτες, ένας για κάθε λίστα, οπότε μπορούμε να χρησιμοποιήσουμε έναν πίνακα δεικτών με δείκτες από "Α" ως "Ω".

Παράδειγμα αλφαβητικού ευρετηρίου κειμένου -1-

- Έστω, ότι έχουμε το παρακάτω κείμενο:
- Ο Κώστας και η Ελένη ζουν σε ένα όμορφο σπίτι στα περίχωρα της πόλης μαζί με τα δυο τους παιδιά, τον Γιάννη και τον Ανέστη.
- Οι παρακάτω συνδεδεμένες λίστες δείχνουν πώς μπορεί να αποθηκευτεί το αλφαβητικό ευρετήριο αυτού του κειμένου:



Παράδειγμα αλφαβητικού ευρετηρίου κειμένου -2-



Για να εμφανιστεί το ευρετήριο αρκεί να διασχίσουμε τις 24 αυτές συνδεδεμένες λίστες.

Αλγόριθμος κατασκευής αλφαβητικού ευρετηρίου -1-

/*Είσοδος: Ένα αρχείο κειμένου.

Λειτουργία: Κατασκευάζει ένα αλφαβητικό ευρετήριο κειμένου από ένα έγγραφο που είναι αποθηκευμένο σε ένα αρχείο. Το ευρετήριο αποθηκεύεται σε μια δομή δεδομένων που αποτελείται από έναν πίνακα συνδεδεμένων λιστών, List, όπου List[Ch] είναι μια ταξινομημένη συνδεδεμένη λίστα λέξεων που ξεκινούν από το γράμμα Ch.

Έξοδος: Μια λίστα των διαφορετικών λέξεων του αρχείου.*/

Αλγόριθμος κατασκευής αλφαβητικού ευρετηρίου -2-

1. Για Ch από 'Α' μέχρι 'Ω'

   Δημιούργησε μια κενή συνδεδεμένη λίστα, List[Ch]

   Τέλος_επανάληψης

2. Άνοιξε το αρχείο και πάρε την πρώτη λέξη Word

Αλγόριθμος κατασκευής αλφαβητικού ευρετηρίου -3-

3. Όσο υπάρχουν λέξεις προς επεξεργασία επανάλαβε
   a) Ψάξε στη λίστα των λέξεων που ξεκινούν από το ίδιο γράμμα που ξεκινά και η Word για να δεις αν ήδη υπάρχει εκεί η λέξη αυτή
   b) Αν δεν υπάρχει η λέξη Word στην αντίστοιχη λίστα των λέξεων τότε εισήγαγε την Word στη λίστα αυτή
   c) Πάρε την επόμενη λέξη Word

   Τέλος_επανάληψης

Αλγόριθμος κατασκευής αλφαβητικού ευρετηρίου -4-

4. Για Ch από 'Α' μέχρι 'Ω'

    Αν η List[Ch] δεν είναι κενή τότε Διάσχισε τη λίστα εμφανίζοντας κάθε λέξη της

    Τέλος_αν

Τέλος_επανάληψης

Υλοποίηση ΑΤΔ Στοίβα και Ουρά με δείκτες

Υλοποίηση στοίβας και ουράς με δείκτες

- Η χρήση πινάκων ως τη βασική αποθηκευτική δομή δεν είναι πιστή υλοποίηση των λιστών, επειδή το σταθερό μέγεθος του πίνακα ορίζει σταθερό μέγεθος για τη λίστα.
- Το ίδιο είδαμε ότι ισχύει και για τις στοίβες και τις ουρές: η υλοποίησή τους με πίνακες θέτει ένα όριο στο μέγεθος της στοίβας ή της ουράς, ενώ θεωρητικά μπορούν να είναι απεριόριστες.
- Τώρα θα δούμε πώς υλοποιούνται οι στοίβες και οι ουρές ως συνδεδεμένες δομές

Υλοποίηση στοίβας με δείκτες -1-

- Μια στοίβα είναι μια λίστα που μπορεί να προσπελαστεί μόνο σε μια άκρη της, την κορυφή.
- Εφόσον, λοιπόν, σε μια συνδεδεμένη λίστα μόνο ο πρώτος κόμβος είναι άμεσα προσπελάσιμος (ο δείκτης List δείχνει πάντα στον πρώτο κόμβο), είναι πολύ λογικό να χρησιμοποιήσουμε μια συνδεδεμένη λίστα για να υλοποιήσουμε μια στοίβα (**linked stack**).

Υλοποίηση στοίβας με δείκτες -2-

- Για μια τέτοια υλοποίηση στοίβας σε C μπορεί να κατασκευαστεί η μονάδα StackADT.c, όπου φαίνονται οι απαραίτητες δηλώσεις καθώς και οι διαδικασίες:
    - δημιουργίας κενής συνδεδεμένης στοίβας,
    - ελέγχου αν μια συνδεδεμένη στοίβα είναι κενή,
    - απώθησης και ώθησης στοιχείου στην συνδεδεμένη στοίβα.
- Οι διαδικασίες:
    - δημιουργίας μιας κενής συνδεδεμένης στοίβας και
    - ελέγχου αν μια συνδεδεμένη στοίβα είναι κενή

  είναι ίδιες με τις αντίστοιχες της συνδεδεμένης λίστας

Υλοποίηση στοίβας με δείκτες -3-

- Η λειτουργία της απώθησης για μια συνδεδεμένη στοίβα είναι στην ουσία η λειτουργία της διαγραφής του πρώτου στοιχείου μιας συνδεδεμένης λίστας, επομένως, η διαδικασία Pop είναι η απλοποιημένη διαδικασία Delete.
- Η διαδικασία της ώθησης σε μια στοίβα είναι παρόμοια με τη διαδικασία της εισαγωγής σε μια συνδεδεμένη λίστα, όταν η εισαγωγή γίνεται στην αρχή της..

Πακέτο για τον ΑΤΔ Συνδεδεμένη Στοίβα -1-

```
// StackADT.h

typedef int
    StackElementType;
     /*ο τύπος των στοιχείων
    της στοίβας ενδεικτικά
    τύπου int */

typedef struct StackNode
    *StackPointer;
```

```
typedef struct StackNode
{
    StackElementType Data;
    StackPointer Next;
} StackNode;

typedef enum {
    FALSE, TRUE
} boolean;
```

Πακέτο για τον ΑΤΔ Συνδεδεμένη Στοίβα -2-

```
void CreateStack(StackPointer *Stack);
boolean EmptyStack(StackPointer Stack);
void Push(StackPointer *Stack,
          StackElementType Item);
void Pop(StackPointer *Stack,
         StackElementType *Item);
```

Πακέτο για τον ΑΤΔ Συνδεδεμένη Στοίβα -3-

```
// StackADT.c

void CreateStack(StackPointer *Stack)
 /*Λειτουργία: Δημιουργεί μια κενή συνδεδεμένη
   στοίβα.
  Επιστρέφει: Μια κενή συνδεδεμένη στοίβα,
   Stack.*/
{
   *Stack = NULL;
}
```

Πακέτο για τον ΑΤΔ Συνδεδεμένη Στοίβα -4-

```
boolean EmptyStack(StackPointer Stack);
/*Δέχεται: Μια συνδεδεμένη στοίβα, Stack.
     Λειτουργία: Ελέγχει αν η Stack είναι κενή.
     Επιστρέφει: TRUE αν η στοίβα είναι κενή,
   FALSE διαφορετικά.*/
{
   return (Stack == NULL);
}
```

Πακέτο για τον ΑΤΔ Συνδεδεμένη Στοίβα -5-

void Pop(StackPointer *Stack,
StackElementType *Item)

/*Δέχεται: Μια συνδεδεμένη στοίβα που η κορυφή της δεικτοδοτείται από τον δείκτη Stack.

Λειτουργία: Αφαιρεί από την κορυφή της συνδεδεμένης στοίβας, αν η στοίβα δεν είναι κενή, το στοιχείο Item.

Επιστρέφει: Την τροποποιημένη συνδεδεμένη στοίβα και το στοιχείο Item.

Έξοδος: Μήνυμα κενής στοίβας, αν η συνδεδεμένη στοίβα είναι κενή.*/

Πακέτο για τον ΑΤΔ Συνδεδεμένη Στοίβα -6-

```
{
    StackPointer TempPtr;
    if (EmptyStack(*Stack))
        printf("EMPTY Stack\n");
   else {
        TempPtr = *Stack; *Item = TempPtr->Data;
        *Stack = TempPtr->Next; free(TempPtr);
    }
}
```

Πακέτο για τον ΑΤΔ Συνδεδεμένη Στοίβα -7-

void Push(StackPointer *Stack,
StackElementType Item)

/*Δέχεται: Μια συνδεδεμένη στοίβα που η κορυφή της δεικτοδοτείται από τον δείκτη Stack και ένα στοιχείο Item.

Λειτουργία: Εισάγει στην κορυφή της συνδεδεμένης στοίβας, το στοιχείο Item.

Επιστρέφει: Την τροποποιημένη συνδεδεμένη στοίβα.*/

Πακέτο για τον ΑΤΔ Συνδεδεμένη Στοίβα -8-

```
{
        StackPointer TempPtr;
        TempPtr = (StackPointer)
        malloc(sizeof(struct StackNode));
        TempPtr->Data = Item;
        TempPtr->Next = *Stack;
    *Stack = TempPtr;
}
```

Πακέτο για τον ΑΤΔ Συνδεδεμένη Ουρά -1-

- Με παρόμοιο τρόπο μπορεί να υλοποιηθεί μια ουρά ως συνδεδεμένη λίστα.
- Η ουρά, είναι μια λίστα, στην οποία αφαιρούνται στοιχεία μόνο από το ένα άκρο της, που λέγεται εμπρός ή κεφάλι, και εισάγονται στοιχεία μόνο στο άλλο άκρο της, το πίσω ή ουρά.
- Σε μια υλοποίηση ουράς ως συνδεδεμένη λίστα (linked queue), λοιπόν, μπορούμε να θεωρήσουμε το πρώτο στοιχείο της λίστας σαν το κεφάλι της ουράς, οπότε η διαδικασία της διαγραφής είναι ίδια με τη διαγραφή από στοίβα.

Πακέτο για τον ΑΤΔ Συνδεδεμένη Ουρά -2-

- Για την εισαγωγή, όμως, στοιχείου στη συνδεδεμένη ουρά θα πρέπει να γίνει διάσχιση της ουράς, ώστε να βρεθεί το τελευταίο στοιχείο της.
- Η διάσχιση μπορεί να αποφευχθεί αν διατηρούμε δύο δείκτες, έναν για το πρώτο στοιχείο, δηλαδή το κεφάλι, και έναν για το τελευταίο, δηλαδή την ουρά της συνδεδεμένης ουράς.

Πακέτο για τον ΑΤΔ Συνδεδεμένη Ουρά -3-

```
// QueueADT.h
typedef int
    QueueElementType;
typedef struct QueueNode
    *QueuePointer;
typedef struct QueueNode
{
    QueueElementType Data;
    QueuePointer Next;
} QueueNode;
```

```
typedef struct
{
    QueuePointer Front;
    QueuePointer Rear;
} QueueType;
typedef enum {
        FALSE, TRUE
} boolean;
```

Πακέτο για τον ΑΤΔ Συνδεδεμένη Ουρά -4-

```
void CreateQ(QueueType *Queue);
boolean EmptyQ(QueueType Queue);
void AddQ(QueueType *Queue,
          QueueElementType Item);
void RemoveQ(QueueType *Queue,
          QueueElementType *Item);
```

Πακέτο για τον ΑΤΔ Συνδεδεμένη Ουρά -5-

```
void CreateQ(QueueType *Queue)
/*Λειτουργία: Δημιουργεί μια κενή συνδεδεμένη
   ουρά.
  Επιστρέφει: Μια κενή συνδεδεμένη ουρά.*/
{
   Queue->Front = NULL;
   Queue->Rear = NULL;
}
```

Πακέτο για τον ΑΤΔ Συνδεδεμένη Ουρά -6-

```
boolean EmptyQ(QueueType Queue)
/*Δέχεται: Μια συνδεδεμένη ουρά.
  Λειτουργία: Ελέγχει αν η συνδεδεμένη ουρά
    είναι κενή.
  Επιστρέφει: TRUE αν η ουρά είναι κενή,
    FALSE  διαφορετικά.*/
{
   return (Queue.Front == NULL);
}
```

Πακέτο για τον ΑΤΔ Συνδεδεμένη Ουρά -7-

```
void RemoveQ(QueueType *Queue,
   QueueElementType *Item)
/*Δέχεται: Μια συνδεδεμένη ουρά.
  Λειτουργία: Αφαιρεί το στοιχείο Item από
    την  κορυφή της συνδεδεμένης
         ουράς, αν δεν είναι  κενή.
  Επιστρέφει: Το στοιχείο Item και την
    τροποποιημένη συνδεδεμένη ουρά.
  Έξοδος:  Μήνυμα κενής ουράς, αν η ουρά
    είναι  κενή.*/
```

Πακέτο για τον ΑΤΔ Συνδεδεμένη Ουρά -8-

```
{
    QueuePointer TempPtr;
    if (EmptyQ(*Queue))  printf("EMPTY Queue\n");
   else {
        TempPtr = Queue->Front; *Item = TempPtr->Data;
        Queue->Front = Queue->Front->Next;
        free(TempPtr);
        if (Queue->Front == NULL)  Queue->Rear = NULL;
    }
}
```

Πακέτο για τον ΑΤΔ Συνδεδεμένη Ουρά -9-

void AddQ(QueueType *Queue,
QueueElementType Item)

/*Δέχεται: Μια συνδεδεμένη ουρά Queue και ένα στοιχείο Item.

Λειτουργία: Προσθέτει το στοιχείο Item στο τέλος της συνδεδεμένης ουράς Queue.

Επιστρέφει: Την τροποποιημένη ουρά.*/

Πακέτο για τον ΑΤΔ Συνδεδεμένη Ουρά -10-

```
{
   QueuePointer TempPtr;
   TempPtr = (QueuePointer) malloc(sizeof(struct
   QueueNode));
   TempPtr->Data = Item; TempPtr->Next = NULL;
   if (Queue->Front == NULL)
         Queue->Front = TempPtr;
   else
         Queue->Rear->Next = TempPtr;
   Queue->Rear=TempPtr;
}
```

Τέλος Ενότητας

Ευρωπαϊκή Ένωση
Ευρωπαϊκό Κοινωνικό Ταμείο
ΕΠΙΧΕΙΡΗΣΙΑΚΟ ΠΡΟΓΡΑΜΜΑ
ΕΚΠΑΙΔΕΥΣΗ ΚΑΙ ΔΙΑ ΒΙΟΥ ΜΑΘΗΣΗ
επένδυση στην κοινωνία της γνώσης
ΥΠΟΥΡΓΕΙΟ ΠΑΙΔΕΙΑΣ ΚΑΙ ΘΡΗΣΚΕΥΜΑΤΩΝ
ΕΙΔΙΚΗ ΥΠΗΡΕΣΙΑ ΔΙΑΧΕΙΡΙΣΗΣ
ΕΣΠΑ 2007-2013
πρόγραμμα για την ανάπτυξη
ΕΥΡΩΠΑΪΚΟ ΚΟΙΝΩΝΙΚΟ ΤΑΜΕΙΟ
Με τη συγχρηματοδότηση της Ελλάδας και της Ευρωπαϊκής Ένωσης