

 **Παράλληλη Επεξεργασία**  
**Κεφάλαιο 10<sup>ο</sup>**  
**Αντίγραφα Εργαζομένων**

Κωνσταντίνος Μαργαρίτης  
Καθηγητής  
Τμήμα Εφαρμοσμένης Πληροφορικής  
Πανεπιστήμιο Μακεδονίας  
[kmarg@uom.gr](mailto:kmarg@uom.gr)  
<http://eos.uom.gr/~kmarg>

Αρετή Καπτάν  
Υποψήφια Διδάκτορας  
Τμήμα Εφαρμοσμένης Πληροφορικής  
Πανεπιστήμιο Μακεδονίας  
[areti@uom.gr](mailto:areti@uom.gr)  
<http://eos.uom.gr/~areti>

# Δυναμική Δημιουργία Εργασιών

- Αλγόριθμοι συνδυαστικής αναζήτησης (πρόβλημα του περιοδεύοντος πωλητή), αλγόριθμοι αναζήτησης γραφημάτων (πρόβλημα συντομότερου μονοπατιού)
- Δυναμική ανάθεση των εργασιών στους επεξεργαστές

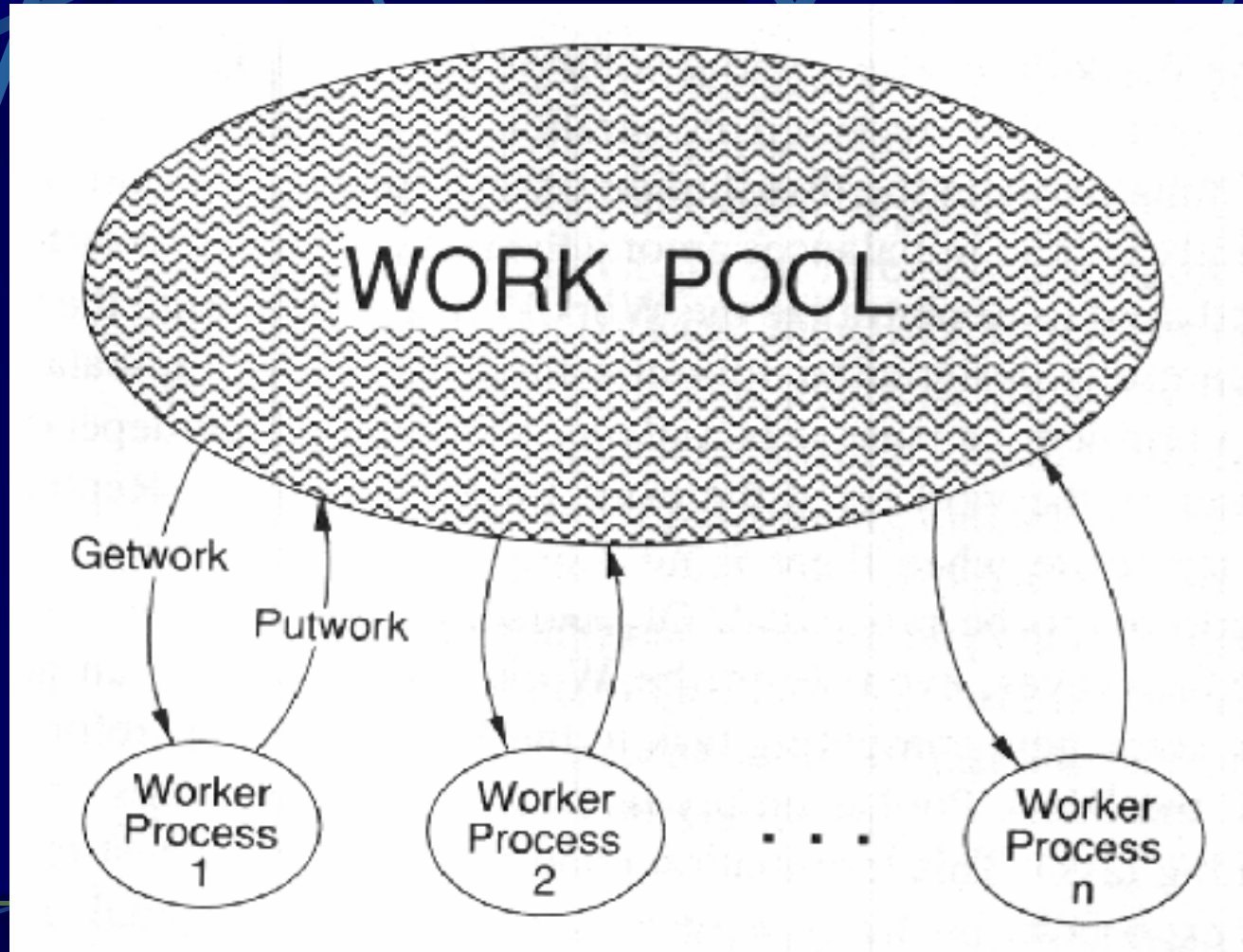
# Αντίγραφα Εργαζομένων σε Παράλληλο Σύστημα Διαμοιραζόμενης Μνήμης

- Διεργασίες-εργαζόμενοι
- Δυναμική ανάθεση υπολογιστικών εργασιών
- Δεξαμενή εργασίας (κανάλια)

# Βασικά χαρακτηριστικά αλγορίθμων Αντιγράφων Εργαζομένων

- Δυναμική δημιουργία νέων υπολογιστικών εργασιών
- Έλεγχος για **συμφόρηση** των καναλιών της δεξαμενής – περιορισμός του αριθμού των διεργασιών
- **Εξισορρόπηση φορτίου** – απασχόληση των διεργασιών-εργαζομένων
- **Τερματισμός** της λειτουργίας των διεργασιών

# Αντίγραφα Εργαζομένων



# Λειτουργικότητα Δεξαμενής Εργασίας

- Άγνωστο πλήθος υπολογιστικών εργασιών
- Κάθε εργαζόμενος μπορεί να δημιουργήσει νέες υπολογιστικές εργασίες

# Υλοποίηση των Αντιγράφων Εργαζομένων σε Π.Σ.Δ.Μ.

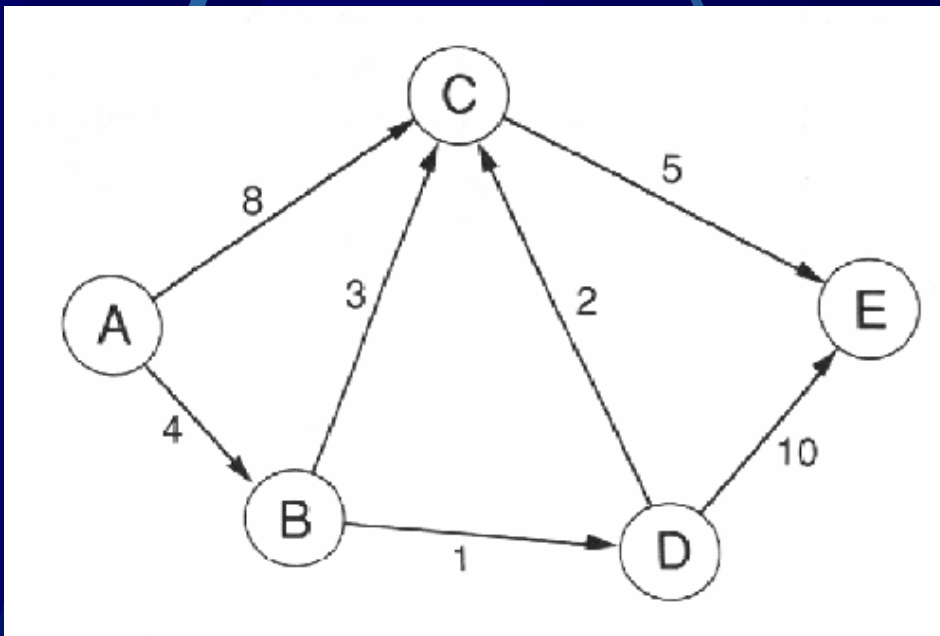
- Δημιουργία διεργασιών-εργαζόμενων με Forall
- Η διεργασία-εργαζόμενος:
  - αποτελείται από ένα βρόχο που καλεί την **GetWork**
  - πιθανότατα καλεί την **PutWork**
  - έχει πρόσβαση σε διαμοιραζόμενες δομές δεδομένων
- Η περιγραφή εργασίας μπορεί να έχει τη μορφή:
  - δομής εγγραφής,
  - πίνακα,
  - απλής τιμής δεδομένων
- Η δεξαμενή εργασίας: υλοποιείται με μεταβλητές-κανάλια

# Θέματα που προκύπτουν κατά τον προγραμματισμό των Αντιγράφων Εργαζομένων

- Συμφόρηση
- Εξισορρόπηση φορτίου
- Μερική αποκέντρωση της δεξαμενής εργασίας
- Τερματισμός:
  - Άδεια δεξαμενή εργασίας
  - Αδρανείς διεργασίες-εργαζόμενοι



# Προσανατολισμένο Γράφημα με Βάρη στις Ακμές



Πίνακας Weight

inf	4	8	inf	inf
inf	inf	3	1	inf
inf	inf	inf	inf	5
inf	inf	2	inf	10
inf	inf	inf	inf	inf

Αρίθμηση Κόμβων: A=1, B=2, C=3, D=4, E=5

# Ακολουθιακός Αλγόριθμος Συντομότερου Μονοπατιού

```
Initialize mindist array to infinity;
Initialize queue to contain source vertex 1;
mindist[1]:= 0;
While queue is not empty do Begin
    x:= head of queue;
    For w:= 1 to n do Begin
        newdist:= mindist[x]+weight[x,w];
        If newdist < mindist[w] then Begin
            mindist[w]:= newdist;
            If w not in queue then append w to queue
        End;
    End;
End;
End;
```

# Παράλληλος Αλγόριθμος Συντομότερου Μονοπατιού

```
PROGRAM Shortpath;
CONST   n=...; (*Αριθμός των κόμβων*)
        numworkers=...; (*Αριθμός των διεργασιών Εργαζομένων*)
        infinity=32000;
TYPE    worktype= INTEGER; (*Κάθε στοιχείο στη Δεξαμενή Εργασίας είναι ένας αριθμός κόμβου*)
VAR     weight: ARRAY [1..n,1..n] OF INTEGER;
        i,j: INTEGER;
        mindist: ARRAY [1..n] OF INTEGER; (*Ελάχιστη απόσταση προς κάθε κόμβο*)
        L: ARRAY [1..n] OF SPINLOCK;
        inflag: ARRAY [1..n] OF BOOLEAN; (*Αληθής αν ο κόμβος βρίσκεται στην Δεξαμενή
Εργασίας*)
        startvertex: worktype;
PROCEDURE Getwork(me: INTEGER; VAR item: worktype);
    ... (*Δέχεται ένα περιγραφέα εργασίας στο "item"*)
PROCEDURE Putwork(me: INTEGER; item: worktype);
    ... (*Πρόσθεση του "item" στη Δεξαμενή Εργασίας*)
```

(...Συνεχίζεται)

# Παράλληλος Αλγόριθμος Συντομότερου Μονοπατιού

```
PROCEDURE Worker(me: INTEGER);                                (...Συνέχεια)
VAR
    vertex: worktype;
    w,newdist: INTEGER;

BEGIN
    Getwork(me, vertex); (*Λαμβάνω έναν νέο αριθμό κόμβου για εξέταση*)
    WHILE vertex <> -1 DO BEGIN
        inflag[vertex]:= FALSE; (*Ο κόμβος αφαιρείται από τη Δεξαμενή Εργασίας*)
        FOR w:= 1 TO n DO BEGIN (*Επεξεργασία όλων των εξερχόμενων ακμών του "vertex"*)
            IF weight[vertex, w] < infinity THEN BEGIN
                (*Ελέγχουμε αν αυτό είναι ένα συντομότερο μονοπάτι προς τον w*)
                newdist:= mindist[vertex]+weight[vertex,w];
                Lock(L[w]); (*Αμοιβαίος αποκλεισμός στο "mindist[w]"*)
                IF newdist < mindist[w] THEN (*Κλείδωμα*) BEGIN
                    mindist[w]:= newdist; (*Ενημέρωση της απόστασης σε "w"*)
                    Unlock(L[w]);
                    IF not inflag[w] THEN BEGIN
                        (*Αν το "w" δεν υπάρχει στη Δεξαμενή Εργασίας*)
                        inflag[w]:= TRUE;
                        Putwork(me,w); (*Τοποθέτηση του "w" στη Δεξαμενή*)
                    END;
                END;
            END;
        END;
    END;
END;
```

(...Συνεχίζεται)

# Παράλληλος Αλγόριθμος Συντομότερου Μονοπατιού

```
(...Συνέχεια)          END
                        ELSE Unlock(L[w]);    (*Εκκλείδωμα*)
                        END; (*IF*)
                    END; (*FOR*)
                Getwork(me,vertex); (*Παίρνω νέο αριθμό κόμβου*)
            END; (*WHILE*)
        END;
    BEGIN (*Κυρίως πρόγραμμα*)
        ... (*Ανάγνωση των τιμών για τον πίνακα weight*)
        FOR i:= 1 TO n DO BEGIN(*Αρχικοποίηση των mindist και inflag*)
            mindist[i]:= infinity;
            inflag[i]:= FALSE;
        END;
        mindist[startvertex]:= 0;
        inflag[startvertex]:= TRUE;
        FORALL i:= 1 TO numworkers DO (*Δημιουργία των Αντιγράφων Εργαζομένων*)
            Worker(i);
            ... (*Τελικές απαντήσεις που βρίσκονται στον πίνακα "mindist"*)
        END.
```

# Συνθήκες Τερματισμού Προγράμματος Αντιγράφων Εργαζομένων (Replicated Workers)

- Η δεξαμενή εργασίας είναι άδεια
- Όλες οι διεργασίες-εργαζόμενοι είναι αδρανείς

# Υλοποίηση των Getwork, Putwork

```
PROGRAM Shortpath;  
CONST   numworkers=...;  
        ...  
TYPE    worktype= INTEGER;  
VAR     workpool: CHANNEL OF worktype;  
        count: INTEGER; (*Μετρητής της Δεξαμενής Εργασίας*)  
        M: SPINLOCK;  
        startvertex: worktype;  
        ...
```

(..Συνεχίζεται)

# Υλοποίηση των Getwork, Putwork

(..Συνέχεια)

```
PROCEDURE Getwork(me: INTEGER; VAR item: worktype);
VAR workcount: INTEGER;
BEGIN
    Lock(M); (*Πρώτη ανάγνωση και μείωση του μετρητή της Δεξαμενής
Εργασίας*)
    workcount:= count-1;
    count:= workcount;
    Unlock(M);
    IF workcount=-numworkers THEN BEGIN (*Τερματισμός των Εργαζομένων*)
        item:= -1;
        FOR i:= 1 TO numworkers-1 DO workpool:= item;
    END
    ELSE item:= workpool; (*Ανάγνωση ενός αντικειμένου από την Δεξαμενή
Εργασίας*)
END;
```

(..Συνεχίζεται)



# Υλοποίηση των Getwork, Putwork

(..Συνέχεια)

```
PROCEDURE Putwork(me: INTEGER; item; worktype);  
VAR workcount: INTEGER;  
BEGIN  
    Lock (M) ;  
    count:= count+1; (*Αύξηση του μετρητή της  
Δεξαμενής Εργασίας*)  
    Unlock (M) ;  
    workpool:= item;  
END;
```

(..Συνεχίζεται)

# Υλοποίηση των Getwork, Putwork

(..Συνέχεια)

```
PROCEDURE Worker(me: INTEGER);
```

```
... (*Διεργασία Worker-δες σχήμα 10.3*)
```

```
BEGIN (*Κυρίως πρόγραμμα*)
```

```
count:= 1;
```

```
startvertex:= 1;
```

```
workpool:= startvertex; (*Ο κόμβος αφειτηρία 1 στη Δεξαμενή  
Εργασίας*)
```

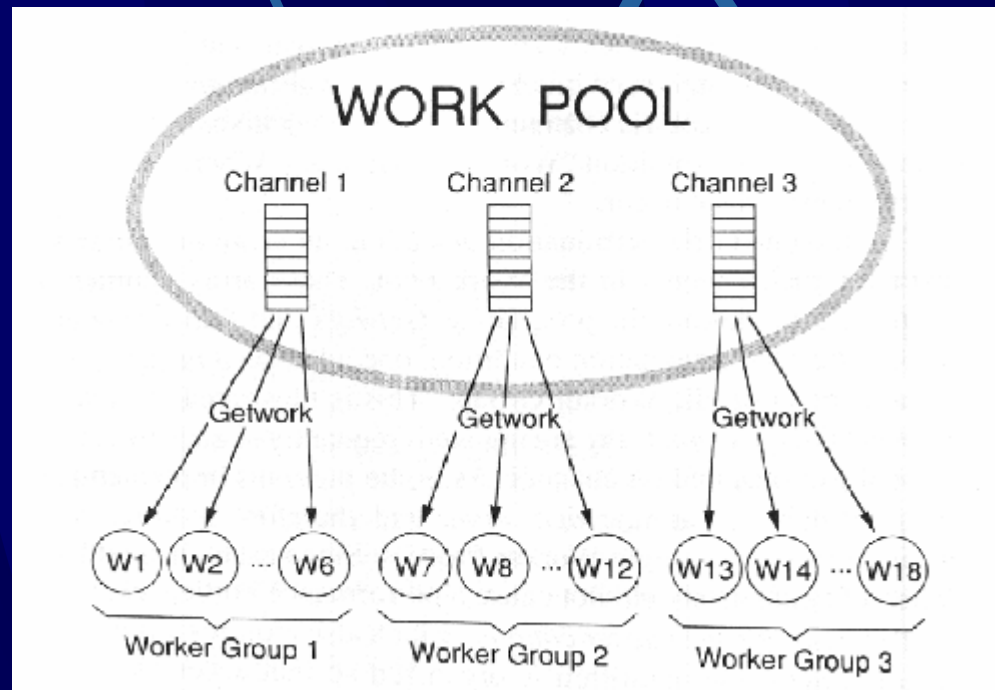
```
... (*Άλλες αρχικοποιήσεις όπως και στο σχήμα 10.3*)
```

```
FORALL i:= 1 TO numworkers DO
```

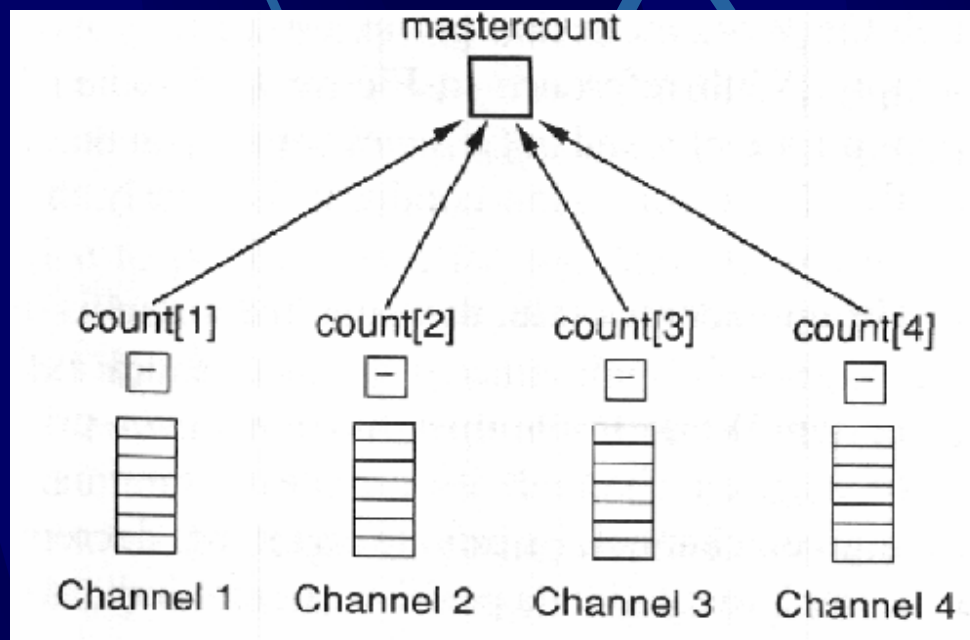
```
    Worker(i);
```

```
END.
```

# Δεξαμενή Εργασίας Πολλαπλών Καναλιών



# Μετρητές Τερματισμού



# Υλοποίηση Πολλαπλών Καναλιών των Δεξ. Εργασίας

```
Getwork(me, item);
Compute my channel number in the Work Pool;
Decrement counter for the channel;
If counter= -Worker_Group_Size then Begin (*Η δική μου Ομάδα Εργαζομένων είναι τώρα αδρανής*)
    Increment master counter;
    If master counter= number_of_Worker_Groups then (*Αποστολή ενός μηνύματος τερματισμού σ
κάθε εργαζόμενο*)
        For i:= 1 to number_of_Worker_Groups do
            For j:= 1 to Worker_Group_Size do
                Put a termination flag into channel of Worker Group i;
        End;
    Read a task descriptor from my channel into "item";

Putwork(me, item)
Move my pointer to the next channel in Work Pool;
Increment counter for the target channel;
If counter= -Worker_Group_Size+1 then
    Decrement master counter; (*Η αδρανής Ομάδα Εργαζομένων είναι τώρα ενεργή*)
Write "item" into the target channel;
```

# Δεξαμενή εργασίας χωρίς συμφύρρηση

```
PROGRAM Shortpath;  
CONST    num_worker_groups= 5;  
         worker_group_size= 10;  
         numworkers= num_worker_groups*worker_group_size;  
TYPE     worktype= INTEGER;  
VAR      workpool: ARRAY [1..num_worker_groups] OF CHANNEL OF worktype;  
         count: ARRAY [1..num_worker_groups] OF INTEGER;  
         CL: ARRAY [1..num_worker_groups] OF SPINLOCK;  
         mastercount: INTEGER;  
         M: SPINLOCK;  
         nextchan: ARRAY [1..numworkers] OF INTEGER;  
         i,j: INTEGER;  
         startvertex: worktype;  
         ...
```

(Συνεχίζεται...)

# Δεξαμενή εργασίας χωρίς συμφόρηση

```
PROCEDURE Getwork(me: INTEGER; VAR item: worktype);
VAR
    worksount, emptycount, mychan: INTEGER;
BEGIN
    mychan:= (me-1) DIV worker_group_size + 1; (*Ο αριθμός καναλιού*)
    Lock(CL[mychan]);
    workcount:= count[mychan] - 1; (*Μείωση του count*)
    count[mychan]:= workcount;
    Unlock(CL[mychan]);
    IF workcount=-worker_group_size THEN BEGIN (*Η Ομάδα Εργαζομένων είναι αδρανής*)
        Lock(M);
        emptycount:= mastercount+1; (*Αύξηση του mastercount*)
        mastercount:= emptycount;
        Unlock(M);
        IF emptycount=num_worker_groups THEN BEGIN (*Τερματισμός όλων των
Εργαζομένων*)
            FOR i:= 1 TO num_worker_groures DO
                FOR j:= 1 TO worker_group_size DO workpool[i]:=
-1;
            END;
        END;
    END;
    item:= workpool[mychan];
END; (*Συνεχίζεται...*)
```

# Δεξαμενή εργασίας χωρίς συμφόρηση

```
PROCEDURE Putwork(me: INTEGER; item: worktype);
VAR workcount, emptycoun, next: INTEGER;
BEGIN
    next:= nextchan[me]; (*Λαμβάνω τον αριθμό του καναλιού προορισμού*)
    Lock(CL[next]);
        workcount:= count[next]+1; (*Αύξηση του count*)
        count[next]:= workcount;
    Unlock(CL[next]);
    IF workcount= -worker_group_size+1 THEN
        BEGIN
            Lock(M);
                emptycount:= mastercount-1; (*Μείωση του mastercount*)
                mastercount:= emptycount;
            Unlock(M);
        END;
    workpool[next]:= item; (*Εγγραφή αντικειμένου στη Δεξαμενή Εργασίας*)
    nextchan[me]:= next MOD num_worker_groups+1; (*Επόμενος προορισμός*)
END;
```

(Συνεχίζεται...)



# Δεξαμενή εργασίας χωρίς συμφόρηση

```
PROCEDURE Worker(me: INTEGER);
VAR ...
BEGIN
    nextchan[me]:= (me-1) DIV worker_group_size+1;
    ... (*Η διεργασία Worker ίδια με αυτή του σχήματος 10.3*)
END;

BEGIN (*Κυρίως πρόγραμμα*)
startvertex:= 1;
workpool[1]:= startvertex; (*Ο κόμβος αφειτηρία 1 μέσα στη
                             Δεξαμενή Εργασίας*)
(*Αρχικοποίηση όλων των αθροιστών*)
count[1]:= 1;
FOR i:= 2 To num_worker_groups DO count[i]:= 0;
mastercount:= 0;

... (*Άλλες αρχικοποιήσεις όπως και στο σχήμα 10.3*)
FORALL i:= 1 TO numworkers DO Worker(i);

END.
```

# Περιοχή Επίθεσης Της Βασίλισσας

		x				x	
		x			x		
x		x		x			
	x	x	x				
x	x	<b>Q</b>	x	x	x	x	x
	x	x	x				
x		x		x			
		x			x		

# Λύση για 7 βασίλισσες

		Q				
						Q
			Q			
Q						
				Q		
	Q					
					Q	

# Περιγραφή του προβλήματος των N βασιλισσών

```
Worker process;
```

```
Getwork(myboard); (*Παίρνω μια νιάδα από την Δεξαμενή Εργασίας*)
```

```
While myboard <> Doneflag Do Begin
```

```
  If myboard.len= n then "A Solution is Found"
```

```
  Else Begin
```

```
    newrow:= len+1;
```

```
    For col:= 1 to n Do Begin
```

```
      Is position (newrow,col) attacked by any of the previous Queens on  
"myboard"?
```

```
      If answer is "no" then Begin
```

```
        create "newboard" by adding Queen in position (newrow,col);
```

```
      End;
```

```
    End;
```

```
    Getwork(myboard);
```

```
  End;
```

```
End;
```

# Πρόγραμμα N Βασιλισσών για Διαμοιραζόμενη Μνήμη

```
PROGRAM Nqueens;  
CONST      n= 8; (*Αριθμός των Βασιλισσών*)  
           num_worker_groups= 5;  
           worker_group_size= 6;  
           numworkers= num_worker_groups*worker_group_size;  
TYPE       board= ARRAY [1..n] OF INTEGER;  
           worktype= RECORD  
               len: INTEGER; (*Αριθμός των τρεχουσών βασιλισσών*)  
               queens: board; (*θέσεις των βασιλισσών*)  
           END;  
VAR        workpool: ARRAY [1..num_worker_groups] OF CHANNEL OF worktype;  
           i: INTEGER; startitem: worktype;  
           solutions: CHANNEL OF board; (*Συλλέγει τις τελικές λύσεις*)  
           ... (*Δηλώσεις αθροιστών της Δεξαμενής Εργασίας όπως και στο σχήμα 10.7*)  
PROCEDURE Getwork(me: INTEGER; VAR item: worktype);  
... (*Ιδία όπως και στο σχήμα 10.7*)  
PROCEDURE Putwork(me: INTEGER; item: worktype);  
... (*Ιδία όπως και στο σχήμα 10.7*)
```

(Συνεχίζεται...)

# Πρόγραμμα N Βασιλισσών για Διαμοιραζόμενη Μνήμη

```
PROCEDURE Worker(me: INTEGER);
VAR      myboard: worktype; row,col,coldist,rowdist: INTEGER;
        samecol,samedia,ok: BOOLEAN;
BEGIN
  Getwork(me,myboard);
  WHILE myboard.len <> -1 DO BEGIN
  WITH myboard DO
    IF len= n THEN solutions:= myboard.queens
    ELSE BEGIN (*Πρόσθεση νέας βασίλισσας στην σκακιέρα*)
      len:= len + 1;
      FOR col:= 1 TO n DO BEGIN
        (*Ελεγχος αν μπορούμε να τοποθετήσουμε την βασίλισσα σε αυτή την στήλη*)
        ok:= true;
        FOR row:= 1 TO len-1 DO BEGIN
          rowdist:= Abs(len-row);
          coldist:= Abs(col-queens[row]);
          samecol:= coldist= 0;
          samedia:= coldist= rowdist;
          IF samecol OR samedia THEN ok:= false;
        END;
```

(Συνεχίζεται...)

# Πρόγραμμα N Βασιλισσών για Διαμοιραζόμενη Μνήμη

```
IF ok THEN BEGIN
    queens[len]:= col;
    Putwork(me, myboard);
    (*Πρόσθεση νέου αντικειμένου στην Δεξαμενή Εργασίας*)
END;
```

```
END;
```

```
END;
```

```
END;
```

```
Getwork(me, myboard); (*Νέο αντικείμενο για εξέταση*)
```

```
END;
```

```
END; (*Worker*)
```

```
BEGIN (*Κυρίως πρόγραμμα*)
```

```
startitem.len:= 0;
```

```
workpool[1]:= statritem; (*Αρχικοποίηση της Δεξαμενής Εργασίας με την άδεια σκακιέρα*)
```

```
... (*Αρχικοποίηση των αθροιστών της Δεξαμενής Εργασίας όπως και στο σχήμα 10.7*)
```

```
FORALL i:= 1 TO numworkers DO Worker(i); (*Δημιουργία των εργαζομένων*)
```

```
... (*Όλες οι απαντήσεις βρίσκονται στο κανάλι "solutions"*)
```

```
END.
```