



Θ.Ε. ΠΛΣ50 (2007-2008) – ΓΡΑΠΤΗ ΕΡΓΑΣΙΑ Ε7

Στόχος

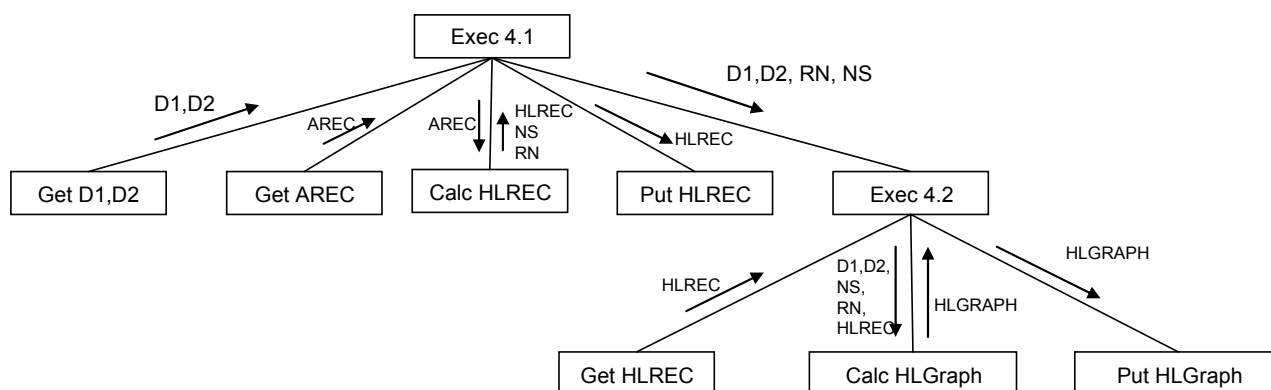
Στην εργασία αυτή θα ασχοληθούμε με θέματα Τεχνολογίας Λογισμικού και συγκεκριμένα με τη λεπτομερή σχεδίαση μονάδων, τα διαγράμματα μετάβασης καταστάσεων και θέματα συνέπειας των προδιαγραφών. Επίσης θα διαπραγματευτεί η έννοια της αντίστροφης μηχανικής και θα τεθούν ορισμένα ζητήματα ελέγχου και διόρθωσης σφαλμάτων του λογισμικού.

ΘΕΜΑ 1: Λεπτομερής σχεδίαση μονάδων.

Δώστε τον ψευδοκώδικα της μονάδας λογισμικού που αντιστοιχεί στη διαδικασία 4.1 (και μόνο αυτής) του διαγράμματος ροής δεδομένων της ενδεικτικής επίλυσης της εργασίας 6.

Απάντηση:

Το ΔΔΠ που αντιστοιχεί στη διεργασία 4.1 του ΔΡΔ έχει ως εξής.



Παρακάτω απεικονίζεται ο ψευδοκώδικας της μονάδας λογισμικού.

PROCEDURE Exec4.1 ()

```
LOCAL VAR D1, D2
LOCAL VAR NS, RN
LOCAL VAR AREC
LOCAL VAR HLREC
Call GetD1D2(D1,D2)
Call GetAREC(AREC)
Call CalcHLREC(AREC,HLREC,NS,RN)
Call StoreHLREC(HLREC)
Call Exec4.2(D1,D2,NS,RN)
END PROCEDURE
```

PROCEDURE Get_D1D2 (D1:IN/OUT, D2:IN/OUT)

```
Open Form on Screen
REPEAT
```



```
READ_FROM_USER D1
      READ_FROM_USER D2
      UNTIL (D1, D2 have the correct format EEEEEMMHH)
END PROCEDURE
PROCEDURE GetAREC (AREC:IN/OUT)
OPEN archivefile
IF (ERROR)
PRINT «ERROR in archivefile »; EXIT
ELSE
REPEAT
AREC += READ AREC
      Until (EOF)
END IF
END PROCEDURE

/* Do two linear scans over all records in the archivefile and identify various
parameters. Then write some of these parameters into a temporary file */
PROCEDURE CalchLREC (AREC:IN/OUT,HLREC:IN/OUT,NS:IN/OUT,RN:IN/OUT)
LOCAL VAR MINDATE, MAXDATE      /* Min,Max in Archive, */
LOCAL VECTOR VSIDS              /* A vector that contains sensor identifiers*/
MINDATE = DATE_MAX              /* Initialize Min date in Archive, DATE_MAX is a
const */
MAXDATE = DATE_MIN              /* Initialize Max date in Archive, DATE_MIN is a
const */
NS = 0                          /* Initialize Number of SensorIDs in Archive
*/
RN = |AREC|                      /* Initialize Number of Measurments in Archive */
/* Find min and max date in archive file as well as number of sensors */
FOR arec FROM 1 TO |AREC| STEP 1 DO
/* Count Number of Sensors */
IF (!contains(arec.SENSOR_ID, VSIDS)) THEN
NS++
END IF
IF (arec.date < MINDATE) THEN /* "<" is an overloaded operator for date
comparisons.*/
MINDATE = arec.date
END IF
IF (arec.date > MAXDATE) THEN /* ">" is an overloaded operator for date
comparisons.*/
MAXDATE = arec.date
END IF
END FOR

LOCAL VAR SIZE      /* SIZE of vector to hold LO,HI temperatures (one for each
date) */
SIZE = MAXDATE - MINDATE
LOCAL VECTOR VLO[SIZE+1] = { DATE_MAX } /* Declaration of a vector to store lo
values */
LOCAL VECTOR VHI[SIZE+1] = { DATE_MIN } /* Declaration of a vector to
store hi values */

/* Now conduct another linear scan in order to define the lo and hi temperature
for each date. */
FOR arec FROM 1 TO |AREC| STEP 1 DO
IF (arec.LO_TEMPVAL < VLO[arec.date]) THEN
VLO[arec.date] = arec.LO_TEMPVAL
END IF
```



```
IF (arec.HI_TEMPVAL > VHI[arec.date]) THEN
VHI[arec.date] = arec.HI_TEMPVAL
    END IF
END FOR
END PROCEDURE

PROCEDURE StoreHLREC (HLREC:IN)
OPEN tempfile
IF (ERROR)
PRINT «ERROR in tempfile »; EXIT
ELSE
FOREACH hrec IN HLREC DO
WRITE tempfile, « hrec.date, hrec.low, hrec.hi newline»
    END FOR
END IF
END PROCEDURE

PROCEDURE Exec4.2 (D1:IN, D2:IN, NS:IN,RN:IN)
LOCAL VAR HLREC
LOCAL VAR HLGRAPH
CALL GetHLRec(HLREC)
CALL CalcHLGRAPH(D1, D2, HLREC, HLGRAPH)
CALL PutHLGRAPH(HLGRAPH)
END PROCEDURE

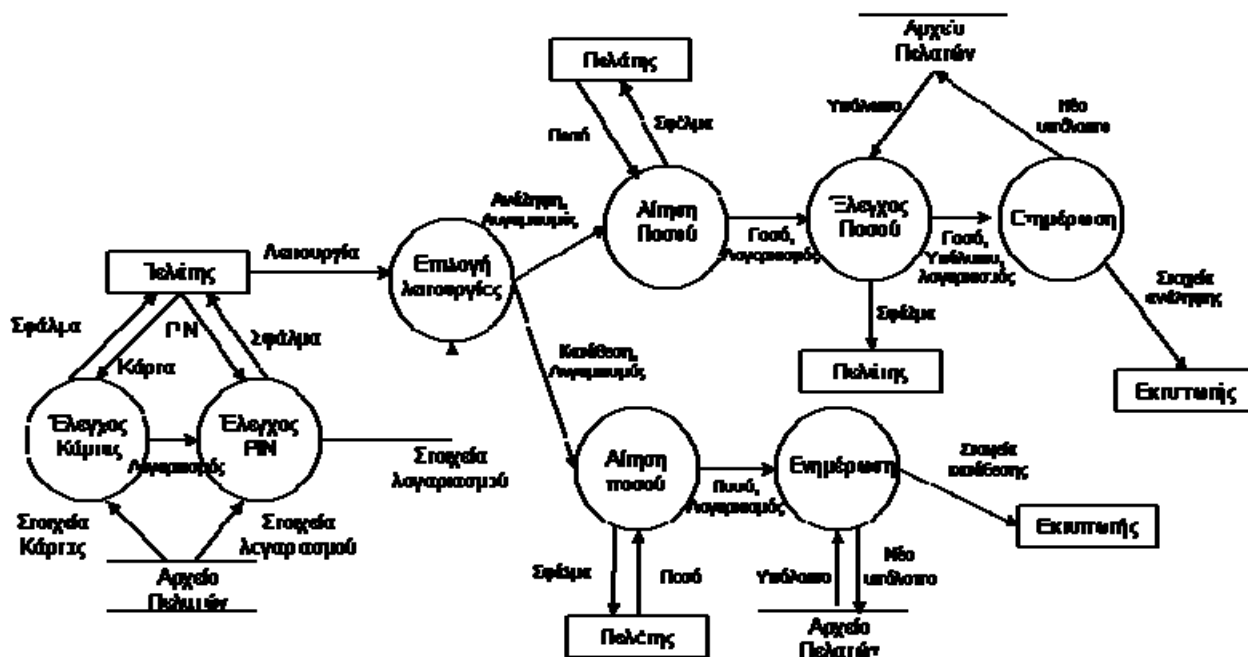
PROCEDURE GetHLRec (HLREC:IN/OUT)
/*Create iterator HREC which can access the tuples of the tempfile one-by-one */
OPEN tempfile
IF (ERROR)
PRINT «ERROR in tempfile »; EXIT
ELSE
REPEAT
HLREC += READ HLREC
    Until (EOF)
END IF
END PROCEDURE

PROCEDURE CalcHLGRAPH (D1:IN, D2:IN, NS:IN, RN:IN, HLREC:IN, HLGRAPH:IN/OUT)
HLGRAPH.NS = NS
HLGRAPH.RN = RN
HLGRAPH.D1 = D1
HLGRAPH.D2 = D2
HLGRAPH.HLREC = HLREC
END PROCEDURE

PROCEDURE PutHLGRAPH (HLGRAPH:IN)
PRINT « NUMOFSENSORS=$HLGRAPH.NS sensors found \newline»
PRINT « NUMOFMEASUREMENTS=$HLGRAPH.RN measurements found \newline»
PRINT « $HLGRAPH.D1/$HLGRAPH.D2 temperature values \newline»
PRINT «date^»
FOR hlgraph FROM 1 TO |HLGRAPH.HLREC| STEP 1 DO
    PRINT « hlgraph.hrec.date | »
PRINT «spaces hlgraph.hrec.lo spaces hlgraph.hrec.hi newline»
END FOR
PRINT «-----+-----»
«-----> value»
END PROCEDURE
```



Έστω το παρακάτω διάγραμμα ροής δεδομένων (ΔΡΔ) που αναπαριστά την απλοποιημένη λειτουργία του λογισμικού ενός ΑΤΜ μίας τράπεζας.



1. Προχωρώντας προς τα «πάνω», απεικονίστε τα *προηγούμενα* επίπεδα λεπτομέρειας του ΔΡΔ μέχρι το πρώτο επίπεδο. Κάντε οποιεσδήποτε παραδοχές κρίνετε σκόπιμες και τεκμηριώστε σε κάθε βήμα την απόφασή σας.

Η προσέγγιση που ακολουθείται για τη δημιουργία των προηγούμενων επιπέδων, βασίζεται στην αναγνώριση υποπεριοχών στο ΔΡΔ οι οποίες μπορούν να βοηθούν ως αυτόνομοι μετασχηματισμοί και οι ροές των δεδομένων που εμπλεκούνται σχετίζονται με συγκεκριμένη λειτουργικότητα. Οι υποπεριοχές αυτές θα αποτελέσουν μια διεργασία σε αμέσως προηγούμενο επίπεδο. Για την αναγνώριση των υποπεριοχών αυτών ακολουθείται μία προσέγγιση bottom-up. Ειδικότερα, οι διεργασίες στο ΔΡΔ και οι εισερχόμενες/εξερχόμενες ροές ταξινομούνται ως προς τη λειτουργικότητά που υπηρετούν.

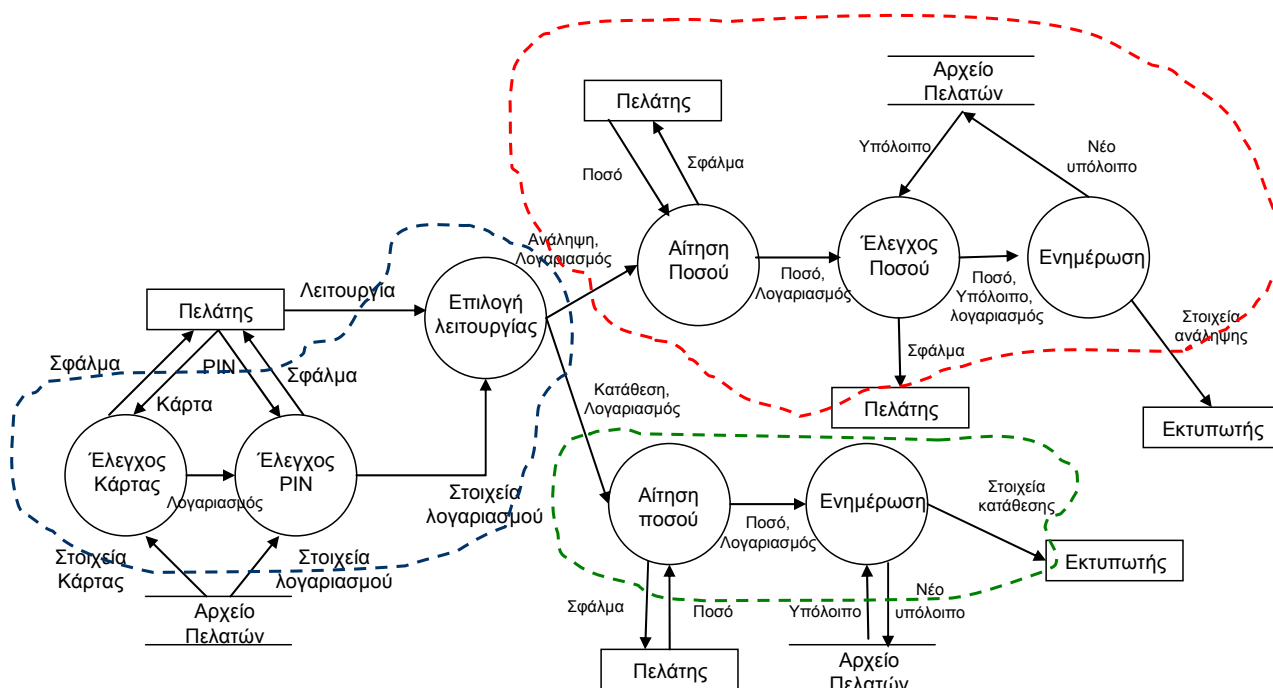
Οι διεργασίες της ίδιας υποπεριοχής πρέπει εμφανίζουν και μία λογική εξάρτηση μεταξύ τους. Αυτό συνήθως σημαίνει, ότι αλλαγή σε μία από αυτές που γίνεται στα πλαίσια της συγκεκριμένης λειτουργικότητας (όπως π.χ. η εισαγωγή νέας εξερχόμενης ροής) ή αλλαγή στην υποπεριοχή (όπως η εισαγωγή νέας διεργασίας) επηρεάζει και τις άλλες διεργασίες της ίδιας υποπεριοχής. Οι



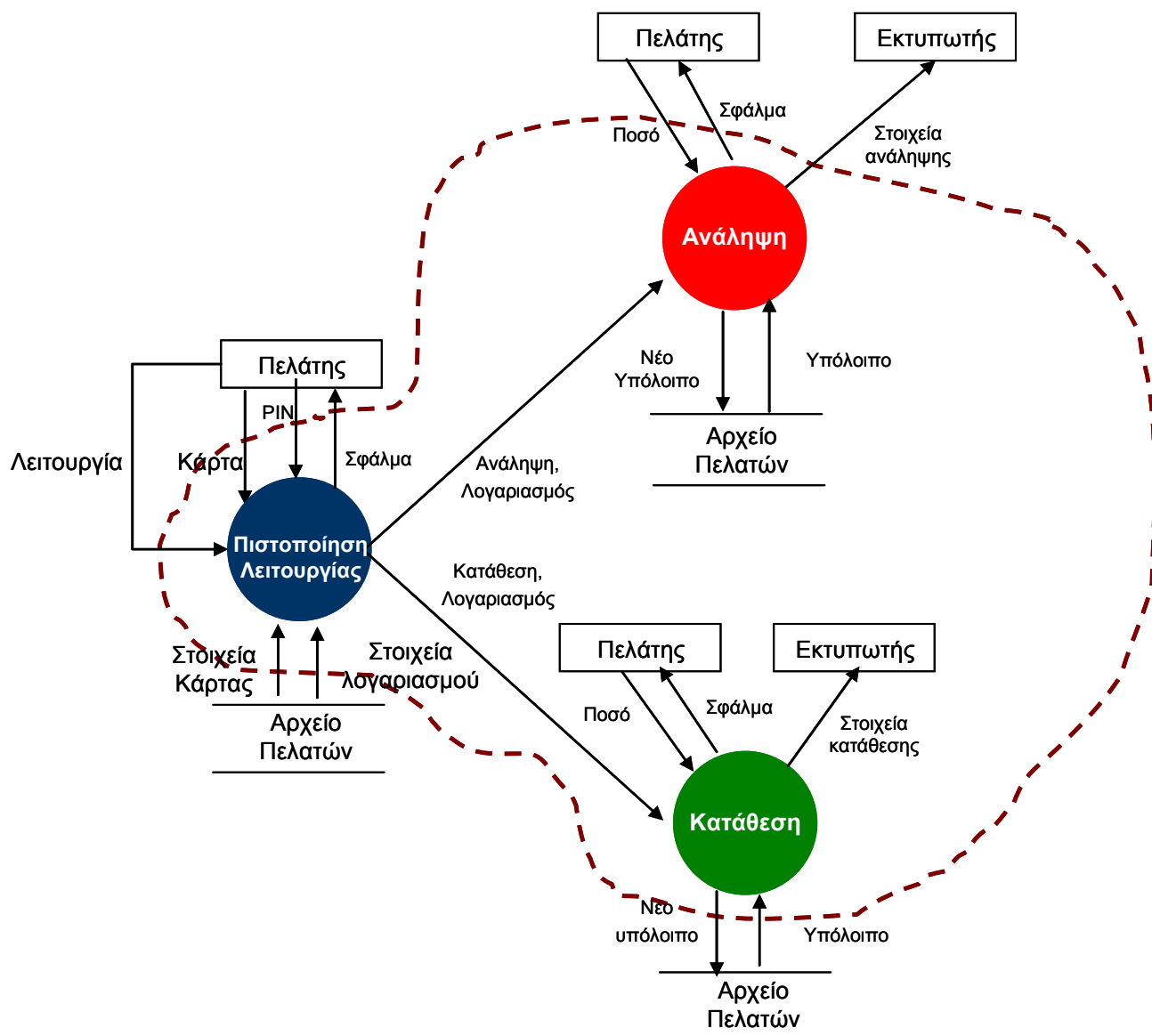
ΕΛΛΗΝΙΚΟ ΑΝΟΙΚΤΟ ΠΑΝΕΠΙΣΤΗΜΙΟ

υποπεριοχές του ΔΡΔ που εμφανίζουν τέτοια εξάρτηση ορίζουν μία «σφαίρα αλλαγής». Συνήθως οι «σφαίρες αλλαγών» παρουσιάζουν μία «κλειστότητα» όσο αφορά τις τροποποιήσεις. Αλλαγές που συντελούνται εντός των υποπεριοχών αυτών δεν έχουν επίπτωση σε άλλες και επηρεάζουν μόνο την υποπεριοχή εντός της οποίας συντελούνται. Αυτή ακριβώς την έννοια έχει ο όρος «αυτόνομος» που αναφέρθηκε παραπάνω για τους μετασχηματισμούς όπως ενσαρκώνονται από τις υποπεριοχές.

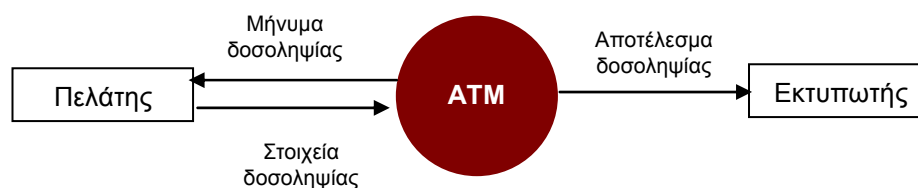
Βήμα 1:



Βήμα 2:



Βήμα 3:

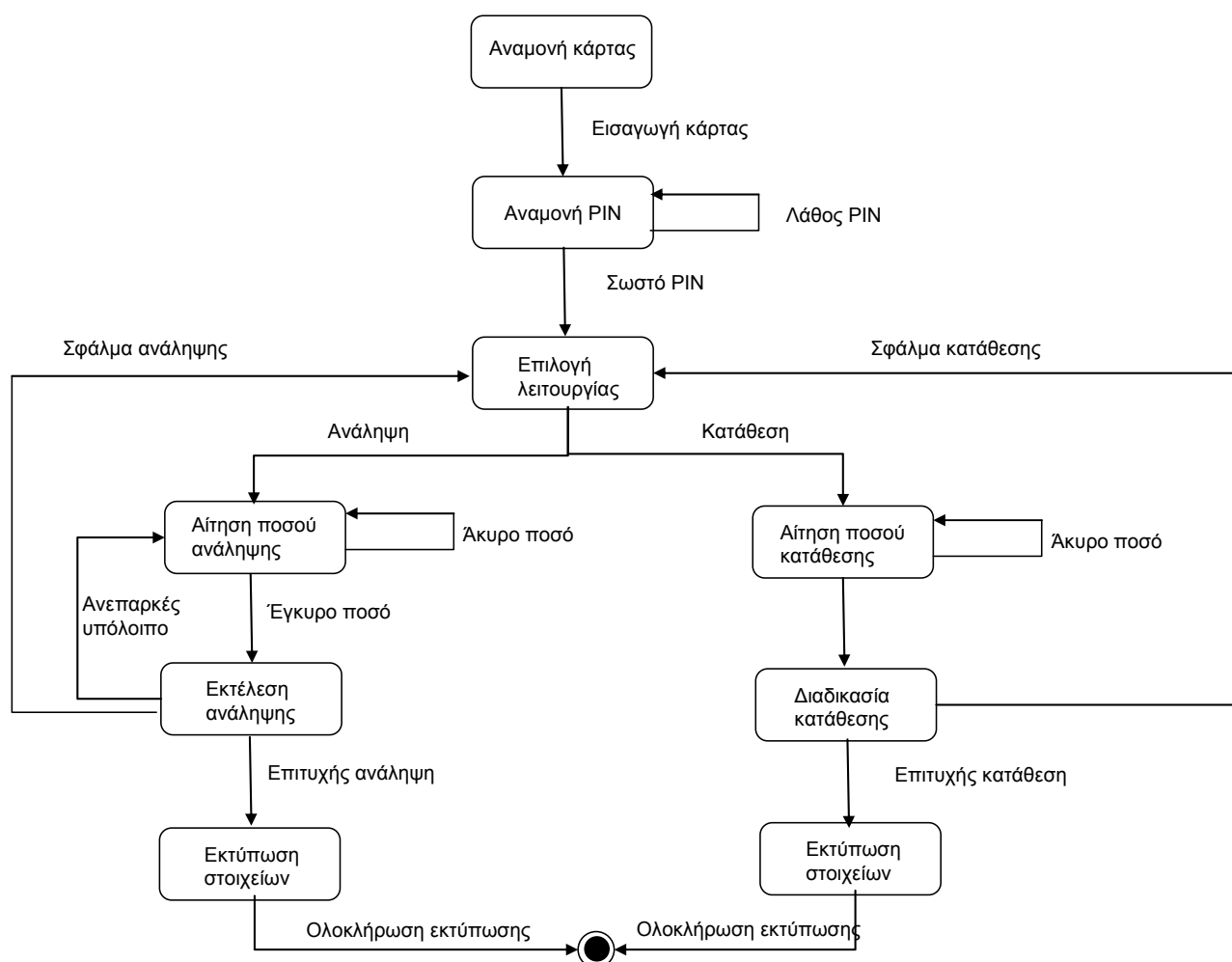




Στο βήμα 3, όπου απεικονίζεται το πρώτο επίπεδο, δεν απεικονίζονται όλες οι ροές εκμεταλλευόμενοι τη δυνατότητα που δίνει το ΔΡΔ οι ροές να αναλύονται σε επόμενα επίπεδα.

2. Φτιάξτε ένα διάγραμμα μετάβασης καταστάσεων (ΔΜΚ) που αντιστοιχεί στο λογισμικό του ΑΤΜ.

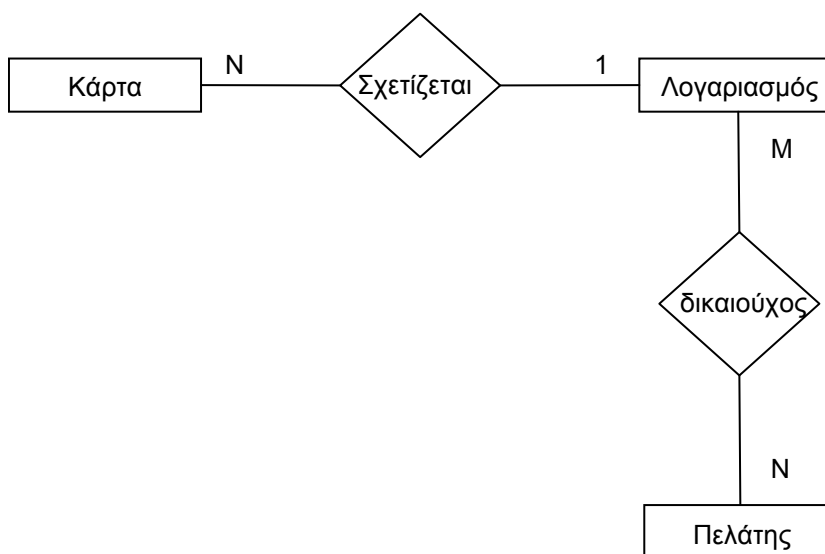
Απάντηση:





3. Στη συγκεκριμένη τράπεζα, κάθε κάρτα είναι συνδεδεμένη με έναν μόνο λογαριασμό ενώ ένας λογαριασμός μπορεί να σχετίζεται με περισσότερες της μίας κάρτες. Οι λογαριασμοί μπορεί να έχουν δικαιούχους πολλούς πελάτες ενώ κάθε πελάτης μπορεί να διατηρεί πολλούς λογαριασμούς. Βάσει των στοιχείων αυτών, κατασκευάστε το διάγραμμα οντοτήτων συσχετίσεων που απεικονίζει το λογισμικό από την οπτική γωνία των δεδομένων της τράπεζας.

Απάντηση:



ΘΕΜΑ 3: Έλεγχος, επαλήθευση και συνέπεια προδιαγραφών.

Για κάθε μία από τις κάτωθι προτάσεις αποφανθείτε αν είναι αληθής ή ψευδής, τεκμηριώνοντας την απάντησή σας.

1. Κάθε διεργασία σε ένα διάγραμμα ροής δεδομένων πρέπει να εμφανίζεται ως κατάσταση σε ένα διάγραμμα μετάβασης καταστάσεων.

Λάθος. Κάθε διεργασία δεν πρέπει υποχρεωτικά να εμφανίζεται ως κατάσταση. Δεν είναι όλες οι διεργασίες επιλέξιμες ως καταστάσεις καθότι πρέπει να πληρούν ορισμένα κριτήρια. Ένα από τα κριτήρια είναι για παράδειγμα αυτό που αναφέρεται στον ορισμό της κατάστασης ότι το σύστημα κάτι αναμένει και «αφουγκράζεται». Επίσης είναι δυνατόν δύο ή περισσότερες διεργασίες να ορίζουν μία κατάσταση.

2. Όπως και στα διαγράμματα ροής δεδομένων (π.χ. βλέπε σχήμα 3.9 του βιβλίου Τεχνολογίας Λογισμικού Ι), έτσι και στα διαγράμματα μετάβασης καταστάσεων μπορούν να υπάρχουν διαδοχικά επίπεδα λεπτομέρειας.



Σωστό. Για παράδειγμα στο βιβλίο ΤΛΙ, σελ. 93, μελέτη περίπτωσης, εμφανίζεται ένα Διάγραμμα Μετάβασης Καταστάσεων που αντιστοιχεί σε μία συγκεκριμένη μονάδα λογισμικού. Με ανάλογο τρόπο μπορούν να δημιουργηθούν και άλλα ΔΜΚ στο ίδιο επίπεδο. Συνεπώς, υπάρχουν διαδοχικά επίπεδα λεπτομέρειας.

3. Τα δεδομένα εξόδου μίας διεργασίας σε ένα διάγραμμα ροής δεδομένων πρέπει να εξαρτώνται μόνο από τα δεδομένα εισόδου αυτής.

Σωστό. Όσο αφορά τις διεργασίες σε ένα Διάγραμμα Ροής Δεδομένων, δεν υπάρχει παρθενογένεση. Νέα δεδομένα δεν μπορούν να δημιουργηθούν «εκ του μηδενός», παρά μόνο με την επεξεργασία όσων εισέρχονται στη διεργασία. Αυτός άλλωστε είναι και ο ρόλος των διεργασιών και από εκεί προέρχεται και το γεγονός ως καλούνται και ως μετασχηματισμοί.

4. Η διαγραφή πληροφορίας από μία αποθήκη σε ένα διάγραμμα ροής δεδομένων συμβολίζεται με μία ροή που εξέρχεται από την εν λόγω αποθήκη.

Λάθος. Συμβολίζεται με μία ροή που εισέρχεται στην αποθήκη. Και αυτό γιατί η διαγραφή επί της ουσίας τροποποιεί και ενημερώνει την αποθήκη.

5. Ένα από τα κριτήρια προκειμένου να συμπεριληφθεί μία κατάσταση σε ένα διάγραμμα μετάβασης καταστάσεων ενός λογισμικού είναι και το εάν αυτή διαρκεί κάποιο πεπερασμένο χρονικό διάστημα (δηλαδή δεν είναι «ακαριαία»).

Σωστό. Όταν λέγεται ότι το λογισμικό βρίσκεται σε μία κατάσταση τότε εννοείται ότι αφουγκράζεται ή αναμένει κάτι και αυτό συνήθως διαρκεί κάποιο χρονικό διάστημα το οποίο είναι αντιληπτό από τον χρήστη. Είναι σημαντικό να τονιστεί ότι το ΔΜΚ επιχειρεί να συλλάβει πως το λογισμικό εξελίσσεται στο χρόνο και γι' αυτό η διάσταση του χρόνου είναι πολύ σημαντική για τα ΔΜΚ. Αυτό είναι σε αντίθεση με το ΔΡΔ, το οποίο δεν διαπραγματεύεται τέτοια θέματα. Έτσι, καταστάσεις που διαρκούν κάποιο χρονικό διάστημα το οποίο μπορεί να γίνει αντιληπτό από τον χρήστη, είναι υποψήφιος ως «καταστάσεις» σε ένα ΔΜΚ. Ωστόσο, αυτό δεν αποτελεί το μόνο κριτήριο. Η έμφαση που δίνει το ΔΜΚ στη διάσταση χρόνος οφείλεται στο ότι τα διαγράμματα αυτά εμφανίστηκαν στα λεγόμενα συστήματα «Πραγματικού Χρόνου» (real time) όπου η απόκριση του συστήματος σε ένα γεγονός υπόκειται σε χρονικούς περιορισμούς.

Κατά τη φάση σχεδιασμού, υποτίθεται ότι η τεχνολογία που υπάρχει διαθέσιμη είναι «τέλεια», που σημαίνει ότι ορισμένες ενέργειες το λογισμικό τις εκτελεί ακαριαία. Για παράδειγμα στο ΔΡΔ του θέματος 2, η διεργασία «Μορφοποίηση» θεωρείται ότι συντελείται ακαριαία, δηλαδή σε μηδενικό χρόνο και έτσι δεν αποτελεί κατάσταση. Το πόσο χρόνο διαρκεί μία κατάσταση (και συνεπώς αν μπορεί να θεωρηθεί ότι διαρκεί κάποιο χρονικό διάστημα ή όχι) είναι στην υπευθυνότητα του αναλυτή και προκειμένου να αποφανθεί γι' αυτό λαμβάνονται υπόψιν και άλλοι παράγοντες.

ΘΕΜΑ 4: Αντίστροφη μηχανική

Με τον όρο «μηχανική» (engineering) εννοείται συνήθως η συστηματική διαδικασία που ξεκινά από τις απαιτήσεις και τη σχεδίαση και καταλήγει στην παραγωγή του πηγαίου και εκτελέσιμου κώδικα του



προγράμματος. Η αντίστροφη διαδικασία, η οποία ξεκινά από τον πηγαίο ή ακόμη και τον εκτελέσιμο κώδικα ενός προγράμματος και τον μελετά προκειμένου να καταλήξει στις απαιτήσεις και τη σχεδίασή του, εμφανίζεται σε διαδικασίες που είναι γνωστές με τον όρο «αντίστροφη μηχανική» (reverse engineering). Στο χώρο της πληροφορικής, ο στόχος της αντίστροφης μηχανικής συνήθως είναι να φτιαχτεί ένα άλλο, νέο πρόγραμμα που έχει την ίδια λειτουργικότητα με ένα δοθέν, δίχως ωστόσο να το αντιγράψει. Η έννοια της αντίστροφης μηχανικής δεν συναντιέται μόνο στο χώρο της πληροφορικής αλλά και σε άλλες περιοχές (π.χ. μηχανολογία -για περισσότερες πληροφορίες βλ. http://en.wikipedia.org/wiki/Reverse_engineering).

Ζητείται το εξής:

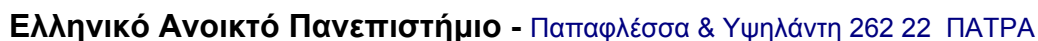
Φτιάξτε ένα διάγραμμα ροής δεδομένων (ΔΡΔ) που να αντιστοιχεί στο πρόγραμμα C της ενδεικτικής επίλυσης της εργασίας 3 του τρέχοντος ακαδημαϊκού έτους. Δώστε ένα μόνο ΔΡΔ, απεικονίζοντας τις απαιτήσεις του λογισμικού στο επίπεδο με τη μεγαλύτερη λεπτομέρεια.

Απάντηση:

Παρακάτω απεικονίζεται ένα ΔΡΔ που αντιστοιχεί στο πρόγραμμα της ενδεικτικής επίλυσης 3. Τονίζεται ξανά, ότι είναι ενδεικτικό και δεν είναι το μοναδικό ορθό.

Κάποιες παρατηρήσεις για την καλύτερη ανάγνωση του διαγράμματος.

1. Το ΔΡΔ σχεδιάστηκε βασιζόμενοι στην παρατήρηση ότι το κεντρικό μενού που σηματοδοτείται στον κώδικα εντός της συνάρτησης `main` από την εντολή `while (choice>0)` αποτελεί ένα Κέντρο Δοσοληψίας (βλ. ΤΛΙ, σελίδα 126, σχήμα 4.17). Αυτό οδήγησε στην εισαγωγή της διεργασίας «Επιλογή Λειτουργίας» στο ΔΡΔ γύρω από την οποία αναπτύχθηκαν οι διάφορες λειτουργίες του προγράμματος.
2. Η αποθήκη με όνομα «Αποθήκη sudoku» χρησιμοποιείται για αναπαράσθησε το σημείο όπου υπάρχουν μόνιμα αποθηκευμένα τα sudoku με τη μορφή αρχείων εντός του συστήματος διαχείρισης αρχείων του υπολογιστή. Σε αυτήν εγγράφονται και τα ενεργά sudoku αφού το πρόγραμμα τερματίσει.
3. Η αποθήκη με όνομα «Ενεργά sudoku» αναπαριστά τη διασυνδεδεμένη λίστα των sudoku, τα οποία φορτώνονται από τα αρχεία («Αποθήκη sudoku») κατόπιν απαίτησης του χρήστη. Οι αποθήκες σε ένα ΔΡΔ δεν προοιδεάζουν υλοποίηση, και έτσι ακόμη και μεταβλητές στην κεντρική μνήμη -αν βέβαια διαφαίνεται να χρησιμοποιούνται από το πρόγραμμα ως αποθηκευτικός χώρος με την ίδια έννοια που χρησιμοποιούνται για παράδειγμα αρχεία- μπορούν να αναπαρασταθούν ως τέτοιες. Επιπλέον, οι αποθήκες σε ένα ΔΡΔ δεν προοιδεάζουν το είδος της (αν θα είναι αρχείο στο σύστημα ή μία βάση δεδομένων) και έτσι μπορεί να είναι και εξειδικευμένες δομές όπως μία διασυνδεδεμένη λίστα, μία σωρός ή ένας πίνακας κατακερματισμού.
4. Αν και η μνήμη έχει αναπαρασταθεί ως αποθήκη «Ενεργά sudoku», η αποθήκη αυτή έχει μία διαφορά από τις άλλες: όταν το πρόγραμμα τερματίζει, τα περιεχόμενά της διαγράφονται. Δεν αποθηκεύει μόνιμα τα sudoku, κάτι που είναι εγγενές γνώρισμα των αποθηκών. Για να αναπαρασταθεί αυτή η προσωρινή φύση της συγκεκριμένης αποθήκης, μόλις το πρόγραμμα τερματίζει τα περιεχόμενά της διαγράφονται, όπως φαίνεται από την εξερχόμενη ροή «Καθαρισμός» από τη διεργασία «Τερματισμός». Η διεργασία αυτή αναλαμβάνει και την μόνιμη αποθήκευση των ενεργών sudoku στην αποθήκη «Αποθήκη sudoku».





στους περιορισμούς που υπάρχουν στα δεδομένα αυτά. Για τη συγκεκριμένη συνάρτηση και ειδικότερα τα δεδομένα εισόδου αυτής (η παράμετρος `femaleName`) ένας από τους περιορισμούς που προκύπτει είναι, το όνομα αυτό να προσδιορίζει φοιτητή γένους θηλυκού. Οπότε τα μέλη της κλάσης ισοδύναμων τιμών είναι όλα τα ονόματα που μπορούν να αποδοθούν σε θηλυκά. Ωστόσο, η κλάση αυτή παρουσιάζει ένα πρόβλημα: δεν μπορεί να οριστούν γι'αυτήν άκρα καθότι το συγκεκριμένο σύνολο δεν μπορεί να διαταχθεί βάσει και μόνο της πληροφορίας ότι τα ονόματα αυτά δίδονται μόνο σε άτομα συγκεκριμένου γένους.

Ωστόσο, από τη διατύπωση του προβλήματος διακρίνεται ένας επιπλέον περιορισμός, όπως αυτός αποτυπώνεται από τη δήλωση του μέλους `name` της δομής `student`. Τα ονόματα των ατόμων γένους θηλυκού δεν μπορεί να είναι μεγαλύτερα από 4 χαρακτήρες. Αυτό μπορεί να αποτελέσει κριτήριο για τη δημιουργία της κλάσης ισοδύναμων τιμών εισόδου. Υποθέτοντας ότι δεν μπορούν να υπάρξουν ονόματα με κανένα χαρακτήρα (δηλαδή με μηδενικό μήκος), η υπόθεση που γίνεται για το μήκος των έγκυρων ονομάτων είναι ότι πρέπει να αποτελούνται από τουλάχιστον έναν χαρακτήρα και από το πολύ τέσσερις.

Όσο αφορά την κλάση ισοδύναμων τιμών εξόδου, αυτή είναι το σύνολο $\{f, -1\}$ σύμφωνα με το όπως έχει δηλωθεί στην εκφώνηση.

Έτσι προκύπτουν οι παρακάτω περιπτώσεις ελέγχου:

Περίπτωση ελέγχου	<code>femaleName</code>	Αναμενόμενα αποτελέσματα
1	Mary	f
2	Maria	-1
3	M	f
4	\0	-1

Η περίπτωση 4 (η μεταβλητή `femaleName` να λαμβάνει την τιμή '\0') εκφράζει την περίπτωση όπου η τιμή βρίσκεται έξω από το ελάχιστο άκρο (ένας χαρακτήρας) και αναφέρεται στην περίπτωση να έχουμε όνομα με μηδενικό μήκος.

2. Για όσες περιπτώσεις η συνάρτηση δεν περνάει τον έλεγχο, αναφέρετε τι τιμή επιστρέφει και εξηγήστε τη συμπεριφορά του προγράμματος.

Η συνάρτηση `enrollFemale` δεν περνάει τους ελέγχους 2 και 4, που σημαίνει ότι δεν επιστρέφει το αναμενόμενο αποτέλεσμα (-1 και στις δύο περιπτώσεις).

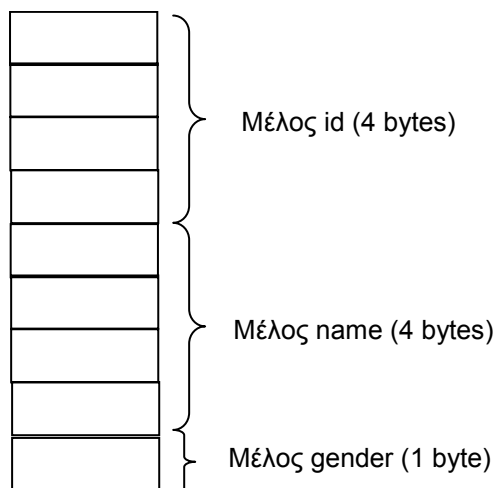
Η περίπτωση ελέγχου 2, που εξετάζει τη περίπτωση το όνομα να είναι έξω από το μέγιστο άκρο δηλαδή ακριβώς 5 χαρακτήρες, δεν επιστρέφει -1 αλλά τον πέμπτο χαρακτήρα της τιμής εισόδου. Έτσι, αν δοθεί ως είσοδο το όνομα 'Maria' θα επιστρέψει 'a' ενώ αν δοθεί ως είσοδο abcde θα επιστρέψει 'e'.

Για να εξηγηθεί η συμπεριφορά αυτή της συνάρτησης πρέπει να δούμε πως τα δεδομένα - και ειδικότερα η δομή `student` - διατάσσονται στη μνήμη.

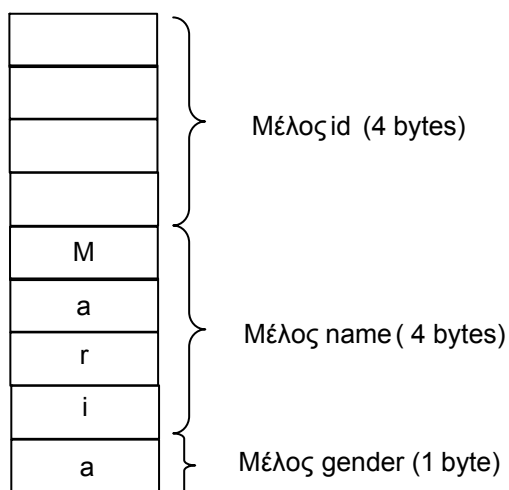
Παρακάτω απεικονίζεται πως η μεταβλητή `s` διατάσσεται στη μνήμη. Όλα τα πεδία αποθηκεύονται σε διαδοχικές θέσεις μνήμης και η ανάγκη για θέσεις μνήμης έχει ως εξής: ο `int` απαιτεί 4 bytes για την αποθήκευσή του ενώ ο `char` 1 byte)



Η μεταβλητή s τύπου student διατάσσεται ως εξής (τα κουτιά αναπαριστούν θέσεις μνήμης):



Όταν με την εντολή strcpy αντιγράφεται η μεταβλητή femaleName στο μέλος name, κάθε χαρακτήρας θα εγγραφεί σε μία θέση μνήμης που έχει δεσμευτεί για το μέλος name. Αν ο αριθμός των χαρακτήρων είναι μεγαλύτερος από 4 (όπως π.χ. στο όνομα Μαρία), η αντιγραφή θα «καταναλώσει» και το χώρο που έχει εκχωρηθεί για το μέλος gender, επειδή οι θέσεις μνήμης έχουν δεσμευτεί διαδοχικά για τη μεταβλητή s. Στην περίπτωση αυτή, οι θέσεις μνήμης θα έχουν τα περιεχόμενα που φαίνονται παρακάτω



Συνεπώς, όταν η enrollFemale θα επιστρέψει την τιμή του μέλους gender, θα επιστραφεί η τιμή 'a' και όχι 'f' ή -1.



Η περίπτωση ελέγχου 2, δεν επιστρέφει τιμή -1 αλλά f.

Αυτό συμβαίνει γιατί η συνάρτηση δεν κάνει κανέναν έλεγχο σχετικά με το ελάχιστο επιτρεπτό μήκος του ονόματος. Ναι μεν η συνάρτηση ελέγχει αν η μεταβλητή `femaleName` έχει τιμή `NULL`, ωστόσο αυτός ο έλεγχος δεν καλύπτει την περίπτωση του μηδενικού μήκους.

Η μεταβλητή `femaleName` είναι δείκτης σε χαρακτήρα και ο έλεγχος

`if (femaleName == NULL)`

εξασφαλίζει αν ο δείκτης παραπέμπει σε έγκυρη θέση μνήμης, ελέγχοντας επί της ουσίας αν έχει δεσμευτεί η απαραίτητη μνήμη για τη τον δείκτη `femaleName`. Δεν ελέγχει το τι υπάρχει αποθηκευμένο στις θέσεις μνήμης όπου παραπέμπει ο δείκτης. Τέτοιος έλεγχος θα απαιτούνταν για να εξεταστεί το μήκος του ονόματος.