HY240: Δομές Δεδομένων
Χειμερινό Εξάμηνο – Ακαδημαϊκό Έτος 2022-23
Διδάσκουσα: Παναγιώτα Φατούρου

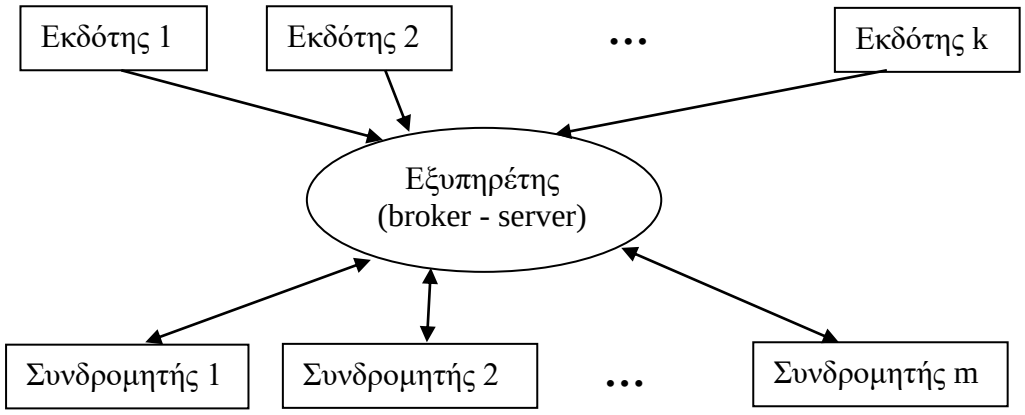
Προγραμματιστική Εργασία
1ο Μέρος

Ημερομηνία Παράδοσης: Δευτέρα, 21 Νοεμβρίου 2022, ώρα 23:59.
Τρόπος Παράδοσης: Χρησιμοποιώντας το πρόγραμμα turnin. Πληροφορίες για το πώς λειτουργεί το turnin παρέχονται στην ιστοσελίδα του μαθήματος.

Γενική Περιγραφή

Σε ένα σύστημα δημοσιοποίησης/συνδρομής (publish subscribe system), οι εκδότες (publishers) είναι παροχείς πληροφοριών ενώ οι συνδρομητές είναι οι καταναλωτές των πληροφοριών αυτών. Οι πληροφορίες που δημοσιεύονται από τους εκδότες κατευθύνονται προς έναν κεντρικό εξυπηρέτη (server) ο οποίος λειτουργεί σαν ενδιάμεσος κόμβος μεταξύ των εκδοτών και των συνδρομητών. Οι πληροφορίες αυτές κατηγοριοποιούνται σε ομάδες (groups) βάσει του περιεχομένου τους ή κάποιων λέξεων κλειδιών που παρέχονται από τους εκδότες.

Κάθε συνδρομητής διατηρεί ένα σύνολο από συνδρομές σε κάποιες από τις ομάδες πληροφοριών που έχουν αποθηκευθεί στον εξυπηρέτη. Κάθε συνδρομητής δηλώνει έτσι το ενδιαφέρον του να λαμβάνει τις πληροφορίες που συσχετίζονται με την εκάστοτε ομάδα. Ο εξυπηρέτης είναι ενήμερος για τις συνδρομές του εκάστοτε συνδρομητή. Το σύστημα που περιγράφεται παραπάνω απεικονίζεται γραφικά στο Σχήμα 1.



Σχήμα 1

Σε ένα σύστημα δημοσιοποίησης/συνδρομής, μπορεί ο εξυπηρέτης να αποθηκεύει τις πληροφορίες που λαμβάνει από τους εκδότες, ή να τις προωθεί απευθείας στους κατάλληλους συνδρομητές. Προκειμένου να είναι εφικτό ένας νέος συνδρομητής να έχει πρόσβαση σε πληροφορίες του παρελθόντος, σε κάποια συστήματα δημοσιοποίησης/συνδρομής, τα οποία ονομάζονται *αναλλοίωτης κατάστασης* (persistent state public/subscribe systems), οι πληροφορίες που λαμβάνονται ή μέρος αυτών σε περίπτωση που ο όγκος τους είναι μεγάλος, αποθηκεύονται στον εξυπηρέτη.

Αναλυτική Περιγραφή Ζητούμενης Υλοποίησης

Στην εργασία αυτή ζητείται να υλοποιηθεί ένα πρόγραμμα που να περιγράφει (προσομοιώνει) τη λειτουργία ενός συστήματος δημοσιοποίησης/συνδρομής αναλλοίωτης κατάστασης (persistent state publish/subscribe system). Πιο συγκεκριμένα, το πρόγραμμα θα πρέπει να υλοποιεί τις ακόλουθες δομές δεδομένων.

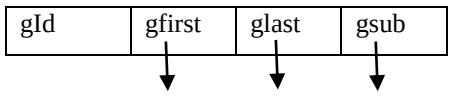
Δομές Δεδομένων που αφορούν δημοσιευμένες πληροφορίες

Όπως προαναφέρθηκε, οι πληροφορίες κατηγοριοποιούνται σε ομάδες (groups). Οι ομάδες που υπάρχουν στο σύστημα αποθηκεύονται σε έναν πίνακα, όπως φαίνεται στο Σχήμα 2, όπου MG (Max_Groups) είναι το μέγιστο πλήθος ομάδων που μπορούν να υπάρχουν στο σύστημα. Ο πίνακας αυτός ονομάζεται *πίνακας ομάδων*.

Σχήμα 2

Κάθε ομάδα αποθηκεύεται ως μια εγγραφή (struct) που αποτελείται από τα πεδία που παρουσιάζονται στη συνέχεια.

- Έναν ακέραιο gId που εκφράζει το αναγνωριστικό της ομάδας. Τα αναγνωριστικά διαφορετικών ομάδων πρέπει να είναι διαφορετικά.
- Δυο δείκτες, gfirst και glast, οι οποίοι δείχνουν στο πρώτο και στο τελευταίο στοιχείο, αντίστοιχα, μιας διπλά συνδεδεμένης, ταξινομημένης, λίστας, η οποία ονομάζεται *λίστα πληροφοριών*. Η λίστα αυτή περιέχει τις πληροφορίες που συσχετίζονται με αυτή την ομάδα.
- Έναν δείκτη (gsub) στο πρώτο στοιχείο μιας απλά συνδεδεμένης λίστας που αποθηκεύει πληροφορίες για τις συνδρομές σε αυτήν την ομάδα και ονομάζεται *λίστα συνδρομών*. Η μορφή κάθε στοιχείου G[i], $0 \leq i \leq MG-1$, του πίνακα ομάδων φαίνεται στο Σχήμα 3.

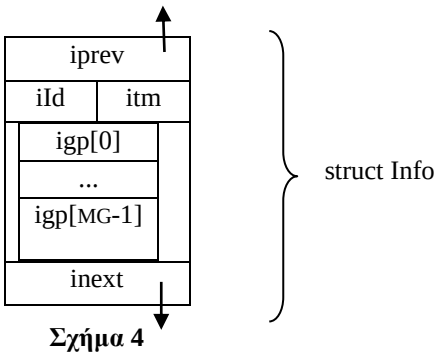


Σχήμα 3

Η λίστα πληροφοριών κάθε ομάδας περιέχει εγγραφές που περιγράφουν τις πληροφορίες που συσχετίζονται με την ομάδα. Κάθε τέτοια πληροφορία αποθηκεύεται ως μια εγγραφή (struct Info) που αποτελείται από τα παρακάτω πεδία.

- Έναν ακέραιο iId που αποτελεί το αναγνωριστικό της πληροφορίας.
- Έναν ακέραιο itm που αποθηκεύει την «χρονική στιγμή» στην οποία η πληροφορία αυτή καταφθάνει στο σύστημα (δηλαδή τη χρονοσφραγίδα άφιξης της πληροφορίας). Η χρονοσφραγίδα άφιξης παρέχεται από το χρήστη όπως περιγράφεται αργότερα. Η λίστα πληροφοριών της εκάστοτε ομάδας είναι ταξινομημένη βάσει του πεδίου itm των εγγραφών της.
- Έναν πίνακα igp[MG] των MG (Maximum_Groups) θέσεων που περιγράφει τα groups πληροφοριών στα οποία ανήκει η πληροφορία. Εάν ισχύει igp[i] == 1, $0 \leq i \leq MG-1$, τότε η πληροφορία έχει συσχετισθεί (ή ανήκει) στην ομάδα πληροφοριών i. Αντίθετα, αν igp[i] == 0, η πληροφορία δεν συσχετίζεται με την ομάδα i.
- Δύο δείκτες iprev και inext που δείχνουν στο επόμενο και στο προηγούμενο στοιχείο της λίστας πληροφοριών στην οποία ανήκει η εγγραφή.

Η εγγραφή τύπου Info παρουσιάζεται στο Σχήμα 4.



Σχήμα 4

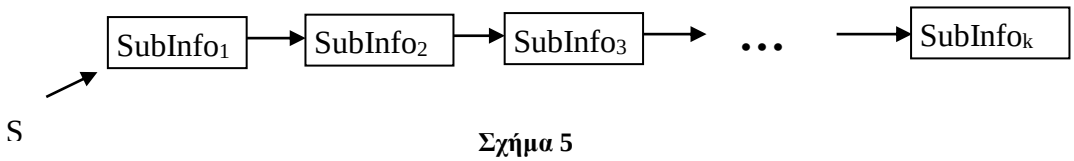
Για κάθε συνδρομή που πραγματοποιείται σε μια ομάδα αποθηκεύεται μια εγγραφή (struct Subscription) στη λίστα συνδρομών της ομάδας αυτής. Η εγγραφή αυτή αποτελείται από τα εξής πεδία:

- έναν ακέραιο sId που αποτελεί το αναγνωριστικό του συνδρομητή που έκανε τη συνδρομή
- έναν δείκτη snext στο επόμενο στοιχείο της λίστας συνδρομών της ομάδας αυτής.

Η μορφή των δομών δεδομένων που περιγράφηκαν παραπάνω παρουσιάζεται στο Σχήμα 8.

Δομές Δεδομένων που αφορούν συνδρομητές

Κάθε συνδρομητής μπορεί να εγγραφεί σε μια ή περισσότερες ομάδες πληροφοριών, ώστε να λαμβάνει τις πληροφορίες που δημοσιεύονται στις ομάδες αυτές. Για κάθε συνδρομητή υπάρχει μια εγγραφή (struct) SubInfo που αποθηκεύει πληροφορίες για το συνδρομητή αυτό. Οι εγγραφές αυτές αποθηκεύονται σε μια απλά συνδεδεμένη λίστα, όπως φαίνεται στο Σχήμα 5. Η λίστα αυτή ονομάζεται *λίστα συνδρομητών*. Παρατηρήστε ότι η λίστα συνδρομητών αποτελεί επιπρόσθετη δομή δεδομένων εκείνων που φαίνονται στο Σχήμα 8.



Σχήμα 5

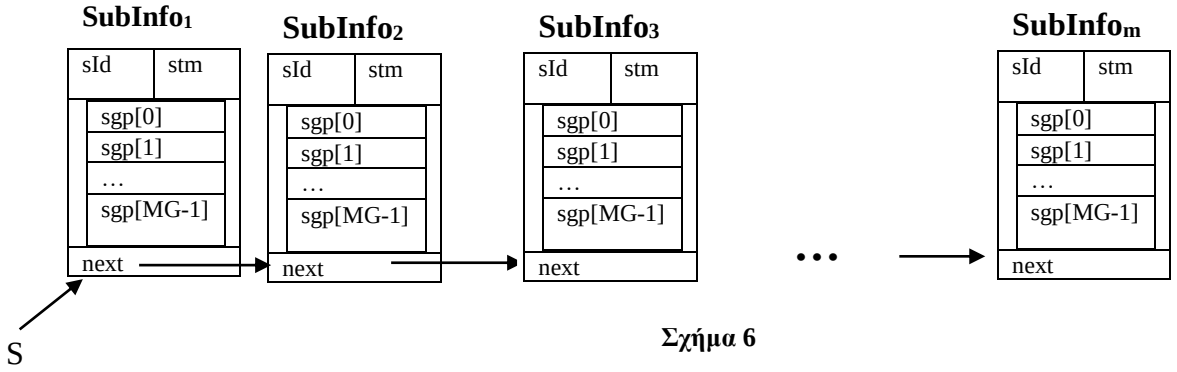
Κάθε εγγραφή SubInfo περιέχει τα εξής πεδία.

- έναν ακέραιο sId που αποτελεί το αναγνωριστικό του συνδρομητή.
- Έναν ακέραιο stm που αποθηκεύει την «χρονική στιγμή» στην οποία ο συνδρομητής καταφθάνει στο σύστημα (δηλαδή τη χρονοσφραγίδα άφιξης του συνδρομητή). Η πληροφορία αυτή παρέχεται από το χρήστη όπως περιγράφεται αργότερα. Η λίστα συνδρομητών είναι ταξινομημένη βάσει του πεδίου stm των εγγραφών της.
- Έναν πίνακα sgp[MG] ο οποίος περιέχει MG δείκτες, έναν για κάθε ομάδα. Έστω οποιαδήποτε ομάδα i, $0 \leq i \leq MG-1$. Αν ο δείκτης sgp[i] μιας εγγραφής της λίστας συνδρομητών έχει την τιμή 1 (μία τιμή που δεν μπορεί να είναι έγκυρη τιμή ενός δείκτη του προγράμματός μας), ο συνδρομητής δεν είναι εγγεγραμμένος στην ομάδα με αναγνωριστικό i. Διαφορετικά, ο δείκτης sgp[i] δείχνει σε ένα από τα στοιχεία της λίστας πληροφοριών της ομάδας G[i]. Περιοδικά, κάθε συνδρομητής μπορεί να καταναλώνει κάποιες από τις πληροφορίες που περιέχονται στις ομάδες στις οποίες είναι συνδρομητής. Συγκεκριμένα, για κάθε ομάδα G[k] καταναλώνει όλες τις πληροφορίες που δημοσιεύθηκαν μετά από την τελευταία εκτέλεση από το συνδρομητή της λειτουργίας κατανάλωσης για την ομάδα αυτή. Αυτό γίνεται με την ανάγνωση των στοιχείων της λίστας πληροφοριών της ομάδας G[k] ξεκινώντας από την εγγραφή sgp[k] (δηλαδή από την τελευταία εγγραφή που ο συνδρομητής έχει καταναλώσει σε αυτή την ομάδα) μέχρι την πιο πρόσφατη εγγραφή που

τοποθετήθηκε στη λίστα πληροφοριών της ομάδας G[k] (δηλαδή την πρώτη εγγραφή της λίστας πληροφοριών της ομάδας). Στο τέλος της διαδικασίας κατανάλωσης, ο δείκτης sgp[k] ενημερώνεται ώστε να δείχνει στο πιο πρόσφατο στοιχείο που περιέχεται στη λίστα πληροφοριών της ομάδας G[k]. Είναι σημαντικό πως ο δείκτης sgp[k] θα έχει την τιμή NULL εάν στην ομάδα G[k] δεν έχει δημοσιευθεί καμία πληροφορία.

- Έναν δείκτη next στο επόμενο στοιχείο της λίστας συνδρομητών.

Αναλυτικά, η μορφή της λίστας συνδρομητών παρουσιάζεται στο Σχήμα 6.



Σχήμα 6

Το πρόγραμμα που θα δημιουργηθεί , θα πρέπει να ορίζει με στατικό τρόπο (με τη χρήση του #define) το μέγιστο πλήθος ομάδων (MG) στο σύστημα και θα πρέπει να εκτελείται καλώντας την ακόλουθη εντολή:

run <input-file>

όπου run είναι το όνομα του εκτελέσιμου αρχείου του προγράμματος και <input-file> είναι το όνομα ενός αρχείου εισόδου που περιέχει γεγονότα των ακόλουθων μορφών:

- I <itm> <iId> <gId_1> <gId_2> … <gId_k> -1: Γεγονός τύπου Insert_Info το οποίο σηματοδοτεί την άφιξη νέας πληροφορίας με αναγνωριστικό <iId> στο σύστημα την χρονική στιγμή <iTM> (η <iTM> αποτελεί τη χρονοσφραγίδα άφιξης της πληροφορίας <iId>). Η πληροφορία θα πρέπει να συσχετισθεί με τις ομάδες <gId_1> <gId_2> … <gId_k>, όπου $0 \leq k \leq MG-1$. Παρατηρήστε ότι η ακολουθία αναγνωριστικών ομάδων με τις οποίες συσχετίζεται η πληροφορία αυτή τερματίζει με το αναγνωριστικό -1 που δεν είναι έγκυρο αναγνωριστικό ομάδας στο σύστημα.

  Οι λίστες πληροφοριών των ομάδων με τις οποίες συσχετίζεται η πληροφορία <iId> πρέπει να ενημερωθούν (δηλαδή θα πρέπει να προστεθεί σε κάθε μια από αυτές μια εγγραφή που να περιγράφει την πληροφορία <iId>). Η εγγραφή αυτή θα πρέπει να εισαχθεί στην κατάλληλη θέση σε κάθε μια από αυτές τις λίστες ώστε αυτή να παραμείνει ταξινομημένη ως προς τη χρονοσφραγίδα άφιξης των πληροφοριών που αποθηκεύονται σε αυτές.

  Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος, το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

  I <iTM> <iId> DONE

  GROUPID = <gId_1>, INFOLIST = <iId_1^1, iId_2^1, …, iId_{r1}^1>

  GROUPID = <gId_2>, INFOLIST = <iId_1^2, iId_2^2, …, iId_{r2}^2>

  …

  GROUPID = <gId_k>, INFOLIST = <iId_1^k, iId_2^k, …, iId_{rk}^k>

  όπου <iId> είναι το αναγνωριστικό της νέας πληροφορίας, <iTM> είναι η χρονοσφραγίδα άφιξης της, ενώ <iId_1^i, iId_2^i, …, iId_{ri}^i>, όπου $1 \leq i \leq k$, είναι τα αναγνωριστικά των εγγραφών πληροφοριών που είναι

αποθηκευμένα στη λίστα πληροφοριών της ομάδας με αναγνωριστικό <gId_i> (όπου έχουμε υποθέσει πως αυτές είναι r_i στο πλήθος).

- S <sTM> <sId> <gId1> <gId2> … <gIdm> -1: Γεγονός τύπου Subscriber_Registration το οποίο σηματοδοτεί την εγγραφή ενός νέου συνδρομητή με αναγνωριστικό <sId> στο σύστημα την χρονική στιγμή <sTM> (η <sTM> αποτελεί τη χρονοσφραγίδα άφιξης του συνδρομητή <sId>). Ο συνδρομητής ενδιαφέρεται για τις πληροφορίες που υπάρχουν στις ομάδες με αναγνωριστικά <gId1> <gId2> … <gIdm>, όπου m ≤ MG. Παρατηρήστε ότι και πάλι η ακολουθία αναγνωριστικών ομάδων για τις οποίες ενδιαφέρεται ο νέος συνδρομητής τερματίζει με το αναγνωριστικό -1 που δεν είναι έγκυρο αναγνωριστικό στο σύστημα.

  Ο νέος συνδρομητής πρέπει να προστεθεί στη λίστα συνδρομητών μαζί με τις πληροφορίες που τον αφορούν. Επίσης πρέπει να προστεθεί μια εγγραφή στη λίστα συνδρομών κάθε ομάδας για την οποία ενδιαφέρεται ο νέος συνδρομητής.

  Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

  S <sTM> <sId> <gId1> <gId2> … <gIdm> DONE

  SUBSCRIBERLIST = <sId_1, sId2, …, sIdm>

  GROUPID = <gId1>, SUBLIST = <sId_1^1, sId_2^1, …, sId_{n1}^1>

  GROUPID = <gId2>, SUBLIST = <sId_1^2, sId_2^2, …, sId_{n2}^2>

  …

  GROUPID = <gIdm>, SUBLIST = <sId_1^m, sId_2^m, …, sId_{nm}^m>

  όπου <sId> είναι το αναγνωριστικό του νέου συνδρομητή, <sTM> είναι η χρονοσφραγίδα άφιξης του συνδρομητή στο σύστημα, <gIdi>, 1 ≤ i ≤ m, είναι οι ομάδες για τις οποίες ενδιαφέρεται ο συνδρομητής αυτός και <sId_1^i, sId_2^i, …, sId_{ni}^i> είναι τα αναγνωριστικά συνδρομητών που είναι αποθηκευμένα στη λίστα συνδρομών της ομάδας <gIdi> (όπου έχουμε υποθέσει πως αυτές είναι n_i στο πλήθος).

- C <sId>: Γεγονός τύπου Consume το οποίο σηματοδοτεί την εκκίνηση της ενέργειας κατανάλωσης των πιο πρόσφατων πληροφοριών κάθε ομάδας για την οποία ενδιαφέρεται ο συνδρομητής με αναγνωριστικό <sId>. Για κάθε ομάδα G[k] για την οποία ενδιαφέρεται ο συνδρομητής <sId>, θα πρέπει να διαβασθούν οι πιο πρόσφατες δημοσιευμένες πληροφορίες (μετά την τελευταία εκτέλεση της λειτουργίας κατανάλωσης του <sId> για το G[k]) μέσω του δείκτη sgp[k] που διατηρεί ο <sId>. Θα πρέπει να εκτυπώνονται τα αναγνωριστικά των πληροφοριών που καταναλώνονται και στο τέλος θα πρέπει να ενημερωθεί ο δείκτης sgp[k] ώστε να δείχνει στην πιο πρόσφατη πληροφορία της ομάδας G[k].

  Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

  C <sId> DONE

  GROUPID = <gId1>, INFOLIST = <iId_1^1, iId_2^1, …, iId_{r1}^1>, NEWSGP = <iId1>

  GROUPID = <gId2>, INFOLIST = <iId_1^2, iId_2^2, …, iId_{r2}^2>, NEWSGP = <iId2>

  …

  GROUPID = <gIdk>, INFOLIST = <iId_1^k, iId_2^k, …, iId_{rk}^k>, NEWSGP = <iIdk>

  όπου <sId> είναι το αναγνωριστικό του νέου συνδρομητή, <gIdi>, όπου $1 \le i \le k$, είναι τα αναγνωριστικά των ομάδων στις οποίες έχει εγγραφεί ο συνδρομητής και <iId_1^i, iId_2^i, …, iId_{ri}^i> είναι

τα αναγνωριστικά των εγγραφών πληροφοριών που καταναλώθηκαν από τη λίστα πληροφοριών της ομάδας G[<gIdi>]. Τέλος, iIdi είναι το αναγνωριστικό της τελευταίας πληροφορίας που καταναλώθηκε από τη λίστα πληροφοριών της ομάδας με αναγνωριστικό <gIdi> (στην οποία θα πρέπει να δείχνει ο δείκτης sgp[<gIdi>]) όταν η λειτουργία κατανάλωσης εκτελεστεί).

- D <sId>: Γεγονός τύπου Delete_Subscriber. Το γεγονός αυτό σηματοδοτεί την αποχώρηση του συνδρομητή με αναγνωριστικό <sId> από το σύστημα. Η εγγραφή που αναφέρεται στον συνδρομητή με αναγνωριστικό <sId> θα πρέπει να διαγραφεί από τη λίστα συνδρομητών. Επίσης, θα πρέπει να διαγραφεί η εγγραφή που αναφέρεται στη συνδρομή του εν λόγω συνδρομητή από κάθε ομάδα στην οποία έχει εγγραφεί ο συνδρομητής.

  Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

  D <sId> DONE

  SUBSCRIBERLIST = <sId_1, sId2, …, sIdm>

  GROUPID = <gId1>, SUBLIST = <sId_1^1, sId_2^1, …, $sIdn_1^1$>

  GROUPID = <gId2>, SUBLIST = <sId_1^2, sId_2^2, …, sId_{n2}^2>

  …

  GROUPID = <gIdk>, SUBLIST = <sId_1^k, sId_2^k, …, sId_{nk}^k>

  όπου <sId> είναι το αναγνωριστικό του νέου συνδρομητή, <gIdi>, 1 <= i <= m, είναι οι ομάδες για τις οποίες ενδιαφέρεται ο συνδρομητής αυτός και sId_1^i, sId_2^i, …, sId_{ni}^i είναι το αναγνωριστικό των στοιχείων της λίστας συνδρομών της ομάδας <gIdi> (όπου έχουμε υποθέσει ότι έχει n_i στοιχεία).

- P: Γεγονός τύπου Print που σηματοδοτεί την εκτύπωση των δομών δεδομένων του συστήματος. Συγκεκριμένα, για κάθε μία από τις ομάδες, θα πρέπει να εκτυπώνεται η λίστα των πληροφοριών της και των συνδρομών της. Επίσης, θα πρέπει να εκτυπώνεται η λίστα με τους συνδρομητές και για κάθε έναν από αυτούς να εκτυπώνονται οι συνδρομές του, δηλαδή εκείνα τα στοιχεία του πίνακα sgp που περιγράφουν τις ομάδες για τις οποίες ενδιαφέρεται ο συνδρομητής.

  Μετά το πέρας της εκτέλεσης ενός τέτοιου γεγονότος το πρόγραμμα θα πρέπει να τυπώνει την ακόλουθη πληροφορία:

  P DONE

  GROUPID = <gId1>, INFOLIST = <iId_1^1, iId_2^1, …, iId_{r1}^1>, SUBLIST = < iId_1^1, iId_2^1, …, iId_{n1}^1>

  GROUPID = <gId2>, INFOLIST = <iId_1^2, iId_2^2, …, iId_{r2}^2>, SUBLIST = < iId_1^2, iId_2^2, …, iId_{n2}^2>

  …

  GROUPID = <gIdMG>, INFOLIST= <iId_1^{MG}, iId_2^{MG}, …, iId_{rk}^{MG}>, SUBLIST= < iId_1^{MG}, iId_2^{MG},…, iId_{nk}^{MG}>

  SUBSCRIBERLIST = <sId1, sId2, …, sIdm>

  SUBSCRIBERID = <sId1>, GROUPLIST = <gId_1^1, …, gId_{v1}^1>

  SUBSCRIBERID = <sId2>, GROUPLIST = <gId_1^2, …, gId_{v2}^2>

  …

  SUBSCRIBERID = <sIdm>, GROUPLIST = <gId_1^m, …, gId_{vm}^m>

  NO_GROUPS = <number of groups>, NO_SUBSCRIBERS = <number of subscribers>

  όπου <gIdi>, $1 \leq i \leq MG$, είναι το αναγνωριστικό της ομάδας G[<gIdi>], <iId_1^i, iId_2^i, …, iId_{ri}^i> είναι η λίστα πληροφοριών της και < iId_1^i, iId_2^i, …, iId_{ni}^i> είναι η λίστα συνδρομών της. Επιπρόσθετα,

<sIdj> είναι το αναγνωριστικό του j-οστού στοιχείου της λίστας συνδρομητών και <gId_1^j, …, gId_{vj}^j> είναι τα αναγνωριστικά των ομάδων για τις οποίες ενδιαφέρεται ο συνδρομητής με αναγνωριστικό <sIdj>. Τέλος, <number of groups> είναι το συνολικό πλήθος των έγκυρων ομάδων και <number of subscribers> είναι το συνολικό πλήθος των συνδρομητών στο σύστημα.

Το πρόγραμμα που θα δημιουργηθεί πρέπει να διαβάζει το αρχείο εισόδου και να εκτελεί με τη σειρά όλα τα γεγονότα που περιγράφονται σε αυτό.

Οδηγίες και Συμβουλές για την Ομαλή Διεκπεραίωση της Εργασίας

Η εργασία θα πρέπει να πραγματοποιηθεί σε βήματα. Ο κάθε φοιτητής είναι υπεύθυνος να αποφασίσει ποια βήματα ταιριάζουν στον τρόπο εργασίας του. Στη συνέχεια, παρατίθεται μια δυνητική ακολουθία βημάτων που θα μπορούσε να ακολουθηθεί για την ομαλή διεκπεραίωση της εργασίας.

Βήμα 1: Ξεκινήστε υλοποιώντας ένα μέρος του γεγονότος S (Subscriber_Registration). Για την υλοποίηση του γεγονότος αυτού πρέπει αρχικά να υλοποιήσετε τη λίστα συνδρομητών. Η λίστα αυτή είναι μια απλά συνδεδεμένη λίστα, που όμως πρέπει να διατηρεί τις εγγραφές ταξινομημένες ως προς το πεδίο stm. Φτιάξτε διαδικασίες SL_Insert, SL_Delete και SL_LookUp για ταξινομημένη συνδεδεμένη λίστα, προκειμένου να πραγματοποιείτε εισαγωγές, διαγραφές και αναζητήσεις, αντίστοιχα, στη λίστα αυτή. Δημιουργήστε μια διαδικασία SL_Print() για την εκτύπωση των στοιχείων της λίστας η οποία θα σας βοηθήσει να ελέγξετε την ορθότητα των διαδικασιών που περιγράφονται παραπάνω.

Δημιουργήστε τον κώδικα σας σε ένα νέο αρχείο και φτιάξτε μια δική σας main() για να ελέγξετε ότι ο κώδικας που υλοποιεί τις λειτουργίες μια απλά-συνδεδεμένης ταξινομημένης λίστας λειτουργεί σωστά. Μόνο όταν ο κώδικας του βήματος 1 εκτελείται σωστά θα πρέπει να προχωρήστε στο βήμα 2.

Βήμα 2: Υλοποιείστε ένα μέρος του γεγονός τύπου I (Insert_Info). Για να γίνει αυτό θα πρέπει να δημιουργήσετε τον κώδικα για μια διπλά συνδεδεμένη ταξινομημένη λίστα που θα αποτελεί τη λίστα των πληροφοριών μιας ομάδας. Όπως και για την απλά-συνδεδεμένη ταξινομημένη λίστα, φτιάξτε διαδικασίες DL_Insert, DL_Delete και DL_LookUp για τη διπλά συνδεδεμένη λίστα, προκειμένου να πραγματοποιείτε εισαγωγές, διαγραφές και αναζητήσεις, αντίστοιχα, στη λίστα αυτή. Δημιουργήστε μια διαδικασία DL_Print() για την εκτύπωση των στοιχείων της λίστας η οποία θα σας βοηθήσει να ελέγξετε την ορθότητα των διαδικασιών που περιγράφονται παραπάνω.

Δημιουργήστε τον κώδικα σας σε ένα νέο αρχείο και φτιάξτε μια δική σας main() για να ελέγξετε ότι ο κώδικας που υλοποιεί τις λειτουργίες μια διπλά-συνδεδεμένης ταξινομημένης λίστας λειτουργεί σωστά. Όταν είστε σίγουροι για αυτό συνεχίστε με το επόμενο βήμα. Ο κώδικας που έχετε δημιουργήσει στο βήμα 1 δεν χρησιμοποιείται καθόλου σε αυτό το βήμα (εδώ απλά υλοποιείτε μια διπλά-συνδεδεμένη ταξινομημένη λίστα της οποίας κάθε στοιχείο είναι τύπου struct info).

Βήμα 3: Ξεκινήστε δημιουργώντας ένα αντίγραφο του βασικού αρχείου (εκείνου που περιέχει τη συνάρτηση main() που σας παρείχαν οι βοηθοί, χωρίς τον κώδικα που φτιάξατε στα βήματα 1 και 2). Στο αρχείο αυτό θα δουλέψετε μόνο με γεγονότα τύπου I. Συγχωνεύστε τον κώδικα που φτιάξατε στο Βήμα 2 στο αρχείο αυτό και κάνετε κατάλληλες αλλαγές ώστε κάθε ομάδα να έχει τη δική της λίστα συνδρομών. Σε κάθε τέτοια λίστα θα πρέπει να μπορούν να εκτελούνται σωστά οι λειτουργίες DL_Insert, DL_Delete και DL_LookUp που δημιουργήσατε στο βήμα 2. Εφόσον είστε βέβαιοι πως οι διαφορετικές λίστες και οι λειτουργίες πάνω σε αυτές λειτουργούν σωστά, προχωρήστε στο επόμενο βήμα. Ο κώδικας που έχετε δημιουργήσει στο βήμα 1 δεν χρησιμοποιείται καθόλου σε αυτό το βήμα (εδώ απλά υλοποιείτε έναν πίνακα του οποίου κάθε στοιχείο περιέχει έναν δείκτη στο πρώτο στοιχείο μιας διπλά συνδεδεμένης ταξινομημένης λίστας με structs τύπου Info).

Βήμα 4: Το βήμα αυτό αποσκοπεί στην υλοποίηση της λίστας των συνδρομών μιας ομάδας. Σε ένα νέο αρχείο, στο οποίο δεν περιέχεται ο κώδικας των βημάτων 1 και 2, γράψτε κώδικα για τις λειτουργίες L_Insert, L_Delete και L_LookUp μιας απλά συνδεδεμένης (μη-ταξινομημένης) λίστας. Επίσης, δημιουργείστε μια διαδικασία L_Print για την εκτύπωση των στοιχείων της λίστας. Ο κώδικας που έχετε δημιουργήσει στα βήματα 1, 2 και 3 δεν χρησιμοποιείται καθόλου σε αυτό το βήμα (εδώ πρέπει να υλοποιήσετε μια απλά συνδεδεμένη λίστα). Χρησιμοποιήστε μια απλή main() (που θα φτιάξετε μόνοι σας) για να ελέγξετε την ορθότητα του κώδικα που φτιάξατε στο βήμα αυτό. Το βήμα αυτό είναι το πιο εύκολο και θα μπορούσατε να ξεκινήσετε με αυτό τη συγγραφή του κώδικάς σας (αντί για το βήμα 1 που περιγράφεται παραπάνω). Σε αυτή την περίπτωση, μετά τη διεκπεραίωση αυτού του βήματος, συνεχίζετε με το τρέχον βήμα 1 και ότι ακολουθεί αμέσως μετά.

Βήμα 5: Μεταφέρετε μέρος του κώδικα του βήματος 4 στο αρχείο που έχετε υλοποιήσει τον κώδικα του βήματος 3 και παράγεται την τελική μορφή του πίνακα G και του κώδικα που απαιτείται για την προσπέλαση και ενημέρωσή του (καθώς και των λιστών που αυτός δεικτοδοτεί). Στο σημείο αυτό μπορείτε να ολοκληρώσετε την υλοποίηση του γεγονότος I προσθέτοντας τη διαδικασία εκτύπωσης των απαραίτητων πληροφοριών που ζητούνται κατά την ολοκλήρωση του γεγονότος αυτού. Η διαδικασία αυτή θα προκύψει συνδυάζοντας τους κώδικες που έχετε δημιουργήσει για την εκτύπωση των στοιχείων των κατάλληλων λιστών.

Βήμα 6: Στο βήμα αυτό θα πρέπει να ολοκληρώσετε την υλοποίηση του γεγονότος S. Συγχωνεύστε τον κώδικα που δημιουργήσατε στο Βήμα 1 και εκείνον του Βήματος 5 για να πάρετε μια πρώτη έκδοση της εργασίας. Στη συνέχεια, προσθέστε τον κώδικα που απαιτείται προκειμένου να εισάγονται οι κατάλληλες εγγραφές στις λίστες συνδρομών των ομάδων για τις οποίες ενδιαφέρεται ο κάθε νέος συνδρομητής. Ετοιμάστε την διαδικασία τύπωσης των στοιχείων που απαιτούνται από το γεγονός τύπου S.

Βήμα 7: Υλοποιήστε το γεγονός τύπου P (Print). Συνδυάστε και τροποποιήστε κατάλληλα τις διαδικασίες SL_Print, DL_Print και L_Print (αρχεία βημάτων 1, 2 και 4) για να δημιουργήσετε τη διαδικασία Print η οποία θα υλοποιεί το γεγονός τύπου P.

Βήμα 8: Υλοποιήστε το γεγονός τύπου C (Consume).

Βήμα 9: Υλοποιήστε το γεγονός τύπου D (Delete_Subscriber). Συνδυάστε κατάλληλα τις διαδικασίες SL_Delete (αρχείο βήματος 1), L_LookUp και L_Delete για να δημιουργήσετε τη διαδικασία Delete που θα υλοποιεί το γεγονός D.

Βήμα 10: Ελέγξετε την ορθότητα του κώδικα που δημιουργήσατε εκτελώντας τον κώδικά σας σε όλα τα αρχεία εκτέλεσης που θα σας παρέχουν οι βοηθοί του μαθήματος. Επιπρόσθετα, δημιουργήστε τα δικά σας αρχεία γεγονότων για να ελέγξετε με περισσότερη ακρίβεια την ορθότητα του κώδικά σας.

Δομές Δεδομένων

Στο Σχήμα 7 παρουσιάζονται οι δομές σε C που πρέπει να χρησιμοποιηθούν για την υλοποίηση της παρούσας εργασίας. Επειδή το πλήθος των ομάδων, δηλαδή το MG, είναι γνωστό εκ των προτέρων πριν την εκτέλεση του προγράμματος, δεσμεύσουμε με στατικό τρόπο τον απαραίτητο χώρο μνήμης για την αποθήκευση των πινάκων που χρησιμοποιούν το MG (G, igp, sgp).

```
struct Info {                       // Εγγραφή λίστας πληροφοριών
        int iId;                    // Αναγνωριστικό πληροφορίας
        int itm;                    // Χρονοσφραγίδα δημοσίευσης πληροφορίας
        int igp[MG];                // Πίνακας igp MG θέσεων
        struct Info *iprev;         // Προηγούμενη εγγραφή λίστας πληροφοριών
        struct Info *inext;         // Επόμενη εγγραφή λίστας πληροφοριών
};

struct Subscription {               // Εγγραφή λίστας συνδρομών
        int sId;                    // Αναγνωριστικό Συνδρομητή
        struct Subscription *snext; // Επόμενη εγγραφή λίστας συνδρομών
};

struct Group {                      // Εγγραφή  πίνακα ομάδων
        int gId;                    // Αναγνωριστικό ομάδας
        struct Subscription *ggsub; // Πρώτο στοιχείο λίστας συνδρομών
        struct Info *gfirst;        // Πρώτο στοιχείο λίστας πληροφοριών
        struct Info *glast;         // Τελευταίο στοιχείο λίστας πληροφοριών
};


struct SubInfo {                    // Εγγραφή λίστας συνδρομητών
        int sId;                    // Αναγνωριστικό συνδρομητή
        int stm;                    // Χρονοσφραγίδα δημιουργίας συνδρομητή
        struct Info *sgp[MG];       // Πίνακας sgp MG θέσεων
        struct SubInfo *snext;      // Επόμενο στοιχείο λίστας συνδρομητών
};
```

```
// Οι παρακάτω δομές θα ήταν προτιμότερο αν δηλωθούν όχι ως global  μεταβλητές
// αλλά ως τοπικές μεταβλητές της main και συνίσταται στους φοιτητές που έχουν
// πολύ καλές γνώσεις προγραμματισμού να ακολουθήσουν αυτή την τακτική
// παρότι η δυσκολία υλοποίησης αυξάνει με αυτό τον τρόπο.

struct Group G[MG];          // Πίνακας G MG θέσεων
struct SubInfo *S;           // Πρώτο στοιχείο λίστας συνδρομητών
```

Σχήμα 7

Bonus

Έχετε τη δυνατότητα να επιβραβευτείτε με κάποιο bonus στο βαθμό σας εάν υλοποιήσετε τις παραλλαγές της εργασίας που παρουσιάζονται στη συνέχεια. Τα bonus είναι ξεχωριστά για κάθε παραλλαγή.

1η Παραλλαγή

Το πρόγραμμα που θα δημιουργηθεί θα πρέπει να εκτελείται καλώντας την ακόλουθη εντολή:

run <m> <input-file>

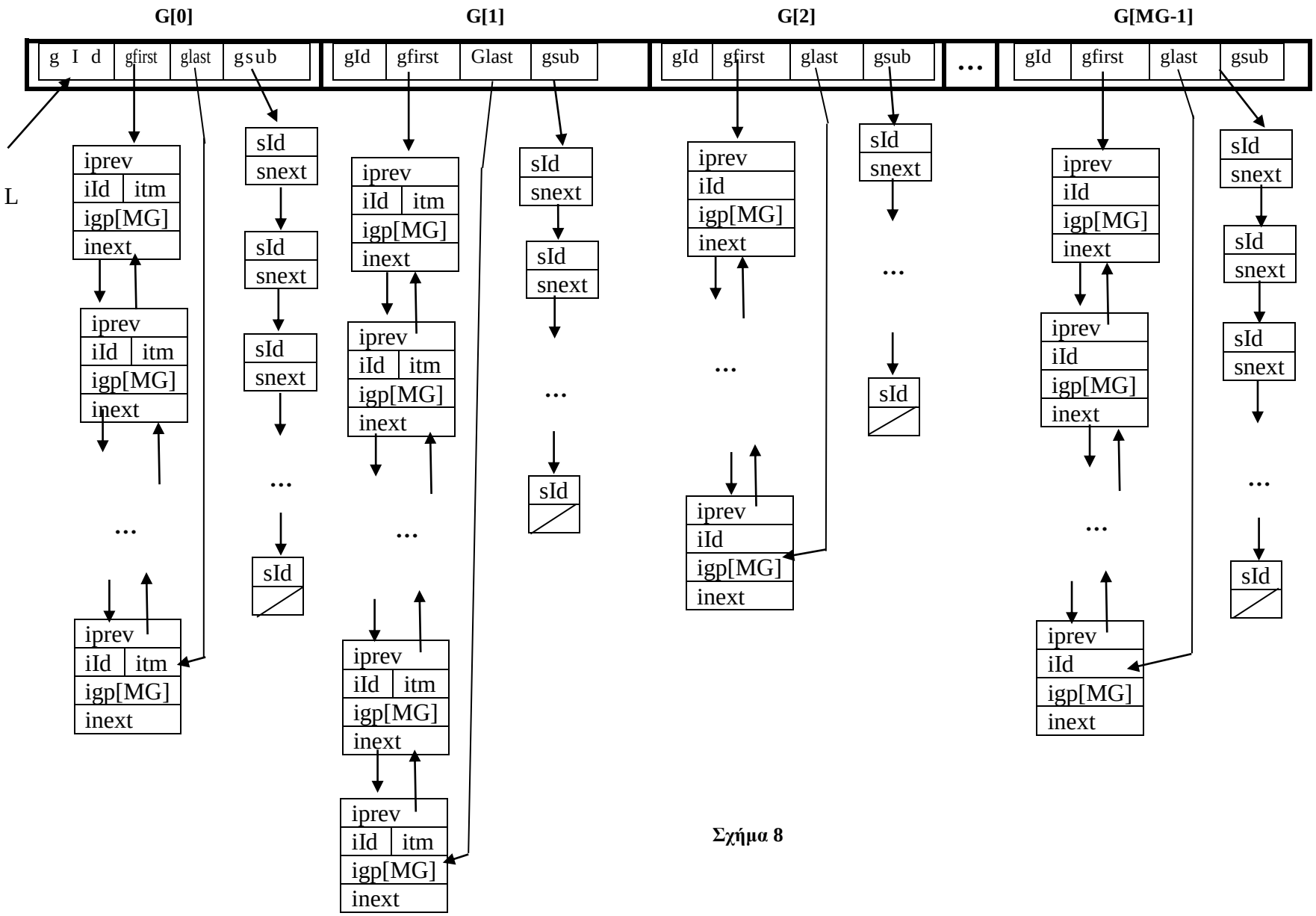
όπου run είναι το όνομα του εκτελέσιμου αρχείου του προγράμματος, <m> είναι ένας ακέραιος που καθορίζει το μέγιστο πλήθος ομάδων (MG) στο σύστημα και <input-file> είναι το όνομα ενός αρχείου εισόδου που περιέχει τα γεγονότα I, S, C, D, P, όπως περιγράφηκαν παραπάνω. Επειδή το πλήθος των ομάδων δεν είναι γνωστό εκ των προτέρων πριν την εκτέλεση του προγράμματος, δεν μπορούμε πλέον να δεσμεύσουμε με στατικό τρόπο τον απαραίτητο χώρο μνήμης για την αποθήκευση των πινάκων που χρησιμοποιούν το MG (G, igp, sgp). Είναι απαραίτητο να δεσμεύεται χώρο μνήμης για τους πίνακες αυτούς κατά την εκτέλεση του προγράμματος, δηλαδή με δυναμικό τρόπο (κατάλληλη χρήση της συνάρτησης malloc).

Η παραλλαγή αυτή επηρεάζει και τη μορφή των δομών δεδομένων που χρησιμοποιούνται στην εργασία. Η νέα μορφή των δομών αυτών παρουσιάζεται στη συνέχεια.

Δομές Δεδομένων 1ης Παραλλαγής

```
struct Info {                    // Εγγραφή λίστας πληροφοριών
        int iId;                 // Αναγνωριστικό πληροφορίας
        int itm;                 // Χρονοσφραγίδα δημοσίευσης πληροφορίας
        int *igp;                // Αυτό πρέπει να είναι ο πίνακας igp MG θέσεων
        struct Info *iprev;      // Προηγούμενη εγγραφή λίστας πληροφοριών
        struct Info *inext;      // Επόμενη εγγραφή λίστας πληροφοριών
};

struct Subscription {            // Εγγραφή λίστας συνδρομών
        int sId;                 // Αναγνωριστικό Συνδρομητή
        struct Subscription *snext;  // Επόμενη εγγραφή λίστας συνδρομών
};

struct Group {                   // Εγγραφή  πίνακα ομάδων
        int gId;                 // Αναγνωριστικό ομάδας
        struct Subscription *ggsub;  // Πρώτο στοιχείο λίστας συνδρομών
        struct Info *gfirst;     // Πρώτο στοιχείο λίστας πληροφοριών
        struct Info *glast;      // Τελευταίο στοιχείο λίστας πληροφοριών
};


struct SubInfo {                 // Εγγραφή λίστας συνδρομητών
        int sId;                 // Αναγνωριστικό συνδρομητή
        int stm;                 // Χρονοσφραγίδα δημιουργίας συνδρομητή
        struct Info **sgp;       // Αυτό πρέπει να είναι ο πίνακας sgp MG θέσεων
```

```
        struct SubInfo *snext;          // Επόμενο στοιχείο λίστας συνδρομητών
};

// Οι παρακάτω δομές θα ήταν προτιμότερο αν δηλωθούν όχι ως global  μεταβλητές
// αλλά ως τοπικές μεταβλητές της main και συνίσταται στους φοιτητές που έχουν
// πολύ καλές γνώσεις προγραμματισμού να ακολουθήσουν αυτή την τακτική
// παρότι η δυσκολία υλοποίησης αυξάνει με αυτό τον τρόπο.

struct Group *G;                        // Αυτό πρέπει να είναι ο πίνακας G, MG θέσεων
struct SubInfo *S;                      // Πρώτο στοιχείο λίστας συνδρομητών
```


G[0]
G[1]
G[2]
G[MG-1]
g I d
gfirst
glast
gsub
gId
Glast
L
iprev
iId
itm
igp[MG]
inext
sId
snext

Σχήμα 8