Περιεχόμενα

[class Message 3](#_Toc9676887)

[Message( string msg, User\* pu) 3](#_Toc9676888)

[int getNumberOfLikes() 3](#_Toc9676889)

[void Like( User \*pUser) 3](#_Toc9676890)

[string toString() 3](#_Toc9676891)

[class ReplyMessage 3](#_Toc9676892)

[ReplyMessage( Message \*pSourceMessage, string msg, User \*pu) 3](#_Toc9676893)

[string toString() 4](#_Toc9676894)

[class User 4](#_Toc9676895)

[User( string na, string em) 4](#_Toc9676896)

[void acceptFriendRequest( FriendRequest \*pRequest); 4](#_Toc9676897)

[string getEmail() 4](#_Toc9676898)

[string getName() 4](#_Toc9676899)

[string getWall() 4](#_Toc9676900)

[void like( string message) 4](#_Toc9676901)

[void postToWall( string message, User \*pSourceUser) 4](#_Toc9676902)

[void postToWall( string message) 4](#_Toc9676903)

[void rejectFriendRequest( FriendRequest \*pRequest) 4](#_Toc9676904)

[void removeFriend( User \*pUser) 4](#_Toc9676905)

[ReplyMessage \*replyToMessage( Message \*pSourceMessage, string message, User \*pSourceUser); 4](#_Toc9676906)

[void sendFriendRequest( User \*pUser) 5](#_Toc9676907)

[string toString() 5](#_Toc9676908)

[class Wall 5](#_Toc9676909)

[list<Message \*> \*getMessagesList() 5](#_Toc9676910)

[void postToWall( Message \*pMessage) 5](#_Toc9676911)

[string toString() 5](#_Toc9676912)

[class FriendRequest 5](#_Toc9676913)

[FriendRequest( User \*pu) 5](#_Toc9676914)

[User \*getUser() 5](#_Toc9676915)

[string toString() 5](#_Toc9676916)

[class Network 5](#_Toc9676917)

[static Network \*getInstance() 5](#_Toc9676918)

[bool areFriends( User \*pUser1, User \*pUser2) 6](#_Toc9676919)

[bool areFriends( string user1, string user2) 6](#_Toc9676920)

[void clear() 6](#_Toc9676921)

[void deleteUser( User \*pUser) 6](#_Toc9676922)

[void deleteUser( string email) 6](#_Toc9676923)

[list<FriendRequest \*> \*getFriendRequestsList( User \*pUser) 6](#_Toc9676924)

[list<User \*> \*getFriendsList( User \*pUser) 6](#_Toc9676925)

[vector<FriendRequest \*> \*getFriendRequestsList( User \*pUser) 6](#_Toc9676926)

[User \*getUser( string email) 6](#_Toc9676927)

[User \*insertUser( string name, string email) 6](#_Toc9676928)

[void makeFriends( User \*pUser1, User \*pUser2) 6](#_Toc9676929)

[void removeFriend( User \*pUser1, User \*pUser2) 6](#_Toc9676930)

[void sendFriendRequest( User \*pUser1, User \*pUser2) 7](#_Toc9676931)

[string toString() { 7](#_Toc9676932)

[bool userExists( string email) 7](#_Toc9676933)

[class Menu 7](#_Toc9676934)

[void AcceptRejectFriendRequest( User \*pUser) 7](#_Toc9676935)

[void DoLike( User \*pUser1, User \*pUser2) 7](#_Toc9676936)

[DoPostMessage( User \*pUser1, User \*pUser2) 7](#_Toc9676937)

[void DoReply( User \*pUser1, User \*pUser2) 7](#_Toc9676938)

[User \*inputUsername() 7](#_Toc9676939)

[void printFriends( User \*pUser) 7](#_Toc9676940)

[User \*selectFriend( User \*pUser) 8](#_Toc9676941)

[void sendFriendRequest( User \*pUser) 8](#_Toc9676942)

[int showMainMenu( User \*pUser) 8](#_Toc9676943)

[int showWallMenu( User \*pUser) 8](#_Toc9676944)

[Εξαιρέσεις 8](#_Toc9676945)

[class UserExistsException 8](#_Toc9676946)

[UserExistsException( string e) 8](#_Toc9676947)

[virtual const char \* what () 8](#_Toc9676948)

[class AlreadyFriendsException 8](#_Toc9676949)

[AlreadyFriendsException (User \*pu1, User \*pu2) 8](#_Toc9676950)

[virtual const char \* what () 8](#_Toc9676951)

[class AlreadyLikeException 8](#_Toc9676952)

[AlreadyLikeException(User \*pu, Message \*pm) 9](#_Toc9676953)

[virtual const char \* what () 9](#_Toc9676954)

[class NoFriendsException 9](#_Toc9676955)

[NoFriendsException (User \*pu1, User \*pu2) 9](#_Toc9676956)

[virtual const char \* what () 9](#_Toc9676957)

[class UserNotExistsException 9](#_Toc9676958)

[class UserNotExistsException( string e) 9](#_Toc9676959)

[virtual const char \* what () 9](#_Toc9676960)

[class SameUserException 9](#_Toc9676961)

[class SameUserException( string e) 9](#_Toc9676962)

[virtual const char \* what () 9](#_Toc9676963)

# class Message

Αναπαριστά τα μηνύματα που γράφονται στον τοίχο.

## Message( string msg, User\* pu)

Ο constructor που δέχεται σαν παραμέτρους το μήνυμα σαν string και το χρήστη ο οποίος το στέλνει.

## int getNumberOfLikes()

Επιστρέφει τον αριθμό των Likes που έχει δεχτεί το συγκεκριμένο μήνυμα.

## void Like( User \*pUser)

Ο χρήστης pUser κάνει Like στο μήνυμα.

Αν έχει ήδη κάνει Like εγείρει εξαίρεση AlreadyLikeException.

## string toString()

Επιστρέφει τα περιεχόμενα σαν string ώστε να εμφανιστούν στην οθόνη σε αναγνώσιμη μορφή.

# class ReplyMessage

Αναπαριστά τις απαντήσεις σε αναρτήσεις του τοίχου.

## ReplyMessage( Message \*pSourceMessage, string msg, User \*pu)

Ο constructor της κλάσης.

## string toString()

Επιστρέφει τα περιεχόμενα σαν string ώστε να εμφανιστούν στην οθόνη σε αναγνώσιμη μορφή.

# class User

## User( string na, string em)

Αρχικοποιεί το χρήστη δίνοντας το όνομα και το email του.

## void acceptFriendRequest( FriendRequest \*pRequest);

Δέχεται το αίτημα φιλίας pRequest.

## string getEmail()

Επιστρέφει το email του χρήστη.

## string getName()

Επιστρέφει το όνομα του χρήστη

## string getWall()

Επιστρέφει τον τοίχο του χρήστη.

## void like( string message)

Κάνει like σε ένα μήνυμα

## void postToWall( string message, User \*pSourceUser)

Στέλνει στον τοίχο το μήνυμα από το χρήστη pSourceUser που μπορεί να είναι είτε ο εαυτός του είτε κάποιος φίλος του.

Αν ο pSourceUser δεν είναι φίλος εγείρει εξαίρεση NoFriendsException.

## void postToWall( string message)

Στέλνει στον τοίχο το μήνυμα από τον ίδιο.

## void rejectFriendRequest( FriendRequest \*pRequest)

Αρνείται το αίτημα φιλίας pRequest.

## void removeFriend( User \*pUser)

Διαγράφει ένα φίλο από τη λίστα των φίλων.

## ReplyMessage \*replyToMessage( Message \*pSourceMessage, string message, User \*pSourceUser);

Γράφει σαν απάντηση στο message μήνυμα του μηνύματος pSourceMessage μηνύματος στον τοίχο του από το χρήστη pSourceUser.

## void sendFriendRequest( User \*pUser)

Δέχεται αίτημα φιλίας από το χρήστη pUser

## string toString()

Επιστρέφει τα περιεχόμενα του χρήστη με μορφή string ώστε να τυπωθούν στην οθόνη.

# class Wall

Αναπαριστά το Wall του χρήστη.

## list<Message \*> \*getMessagesList()

Επιστρέφει λίστα με τα μηνύματα που είναι γραμμένα στον τοίχο.

## void postToWall( Message \*pMessage)

Κάνει post ένα μήνυμα

## string toString()

Επιστρέφει τα περιεχόμενα του τοίχου με μορφή string ώστε να τυπωθούν στην οθόνη.

# class FriendRequest

Αποθηκεύει πληροφορίες για το αίτημα φιλίας.

## FriendRequest( User \*pu)

Ο constructor που δέχεται σαν παράμετρο το χρήστη που έστειλε το αίτημα φιλίας.

## User \*getUser()

Επιστρέφει το χρήστη που έστειλε το αίτημα φιλίας.

## string toString()

Επιστρέφει τα περιεχόμενα αιτήματος φιλίας με μορφή string ώστε να τυπωθούν στην οθόνη.

# class Network

## static Network \*getInstance()

Στην πρώτη φορά που καλείται δημιουργεί ένα αντικείμενο της κλάσης Network και επιστρέφει pointer σε αυτό. Στις επόμενες κλήσεις επιστρέφει τον ίδιο pointer.

## bool areFriends( User \*pUser1, User \*pUser2)

Επιστρέφει true ή false ανάλογα με το αν δύο χρήστες pUser1 και pUser2 είναι φίλοι.

## bool areFriends( string user1, string user2)

Επιστρέφει true ή false ανάλογα με το αν δύο email αντιστοιχούν σε χρήστες που είναι φίλοι.

## void clear()

Διαγράφει όλα τα περιεχόμενα του δικτύου.

## void deleteUser( User \*pUser)

Διαγράφει ένα χρήστη δίνοντας τον pointer σε αυτό το χρήστη.

## void deleteUser( string email)

Διαγράφει ένα χρήστη δίνοντας το email του.

Αν το email δεν αντιστοιχεί σε χρήστη εγείρει την εξαίρεση UserNotExistsException.

## list<FriendRequest \*> \*getFriendRequestsList( User \*pUser)

Επιστρέφει τη λίστα με τα αιτήματα φιλίας προς ένα χρήστη pUser.

## list<User \*> \*getFriendsList( User \*pUser)

Επιστρέφει τη λίστα των φίλων ενός χρήστη pUser

## vector<FriendRequest \*> \*getFriendRequestsList( User \*pUser)

Επιστρέφει τη λίστα με τα αιτήματα φιλίας προς το χρήστη pUser.

## list<User \*> \*getCommonFriends( User \*pUser1, User \*pUser2)

Επιστρέφει λίστα με τους κοινούς χρήστες των pUser1 και pUser2.

## User \*getUser( string email)

Δίνοντας το email επιστρέφει το χρήστη στον οποίο αντιστοιχεί αυτό το email.

Αν το email δεν αντιστοιχεί σε χρήστη επιστρέφει NULL.

## User \*insertUser( string name, string email)

Προσθέτει ένα καινούριο χρήστη στο δίκτυο παίρνοντας σαν παραμέτρους το όνομα και το email του.

## void makeFriends( User \*pUser1, User \*pUser2)

Κάνει φίλους τους χρήστες pUser1 και pUser2.

Αν οι δύο δείκτες αντιστοιχούν στον ίδιο χρήστη εγείρει την εξαίρεση SameUserException.

Αν είναι ήδη φίλοι εγείρει την εξαίρεση AlreadyFriendsException.

## void removeFriend( User \*pUser1, User \*pUser2)

Αφαιρεί από φίλους τους χρήστες pUser1 και pUser2.

Αν δεν είναι φίλοι εγείρει την εξαίρεση NoFriendsException.

## void sendFriendRequest( User \*pUser1, User \*pUser2)

Στέλνει αίτημα φιλίας ο χρήστης pUser1 στο χρήστη pUser2

Αν είναι ήδη φίλοι εγείρει την εξαίρεση AlreadyFriendsException.

## string toString() {

Επιστρέφει με μορφή string τα περιεχόμενα του δικτύου.

## bool userExists( string email)

Επιστρέφει true αν το email αντιστοιχεί σε χρήστη που υπάρχει.

## 

# class Menu

Αναπαριστά το μενού του προγράμματος.

## void AcceptRejectFriendRequest( User \*pUser)

Εμφανίζει στην οθόνη τα αιτήματα φιλίας προς τον pUser και ζητάει αρχικά την επιλογή ενός από αυτά. Στη συνέχεια ζητάει αν θα γίνει αποδοχή ή απόρριψη του αιτήματος φιλίας.

## void DoLike( User \*pUser1, User \*pUser2)

Εμφανίζει τη λίστα των μηνυμάτων του pUser2 και ζητάει την επιλογή ενός μηνύματος ώστε ο χρήστη pUser1 να κάνει Like σε αυτό.

## Επιστρέφει true αν θα πρέπει να πάει ένα βήμα πίσω

## DoPostMessage( User \*pUser1, User \*pUser2)

Γράφει ένα μήνυμα στον τοίχο του χρήστη pUser2 από το χρήστη pUser1.

## void DoReply( User \*pUser1, User \*pUser2)

Εμφανίζει τη λίστα των μηνυμάτων του pUser2 και ζητάει την επιλογή ενός από αυτά.

Στη συνέχεια ζητάει το κείμενο ώστε να το ποστάρει σαν απάντηση ο pUser1 στον τοίχο του pUser2.

## User \*inputUsername()

Ζητάει την πληκτρολόγηση του email και επιστρέφει το χρήστη στον οποίο αντιστοιχεί το email αυτό. Αν το email που πληκτρολογηθεί δεν αντιστοιχεί σε χρήστη ξαναζητάει το email.

## void printFriends( User \*pUser)

Τυπώνει τους φίλους του pUser.

## User \*selectFriend( User \*pUser)

Εμφανίζει στην οθόνη τους φίλους του χρήστη pUser και ζητάει την επιλογή ενός από αυτούς.

## void sendFriendRequest( User \*pUser)

Εμφανίζει όσους δεν είναι φίλοι με τον pUser ή δεν του έχουν στείλει αιτήματα φιλίας ή δεν έχει στείλει αιτήματα φιλίας και ζητάει την επιλογή ενός από αυτούς ώστε να του σταλεί αίτημα φιλίας.

## int showMainMenu( User \*pUser)

Εμφανίζει το κεντρικό μενού ενώ στην πρώτη γραμμή εμφανίζει το email του χρήστη που έχει κάνει “login"

## int showWallMenu( User \*pUser)

Εμφανίζει το μενού με τις επιλογές που αφορούν τον τοίχο (post, reply, like).

# Εξαιρέσεις

## class UserExistsException

Exception για την περίπτωση που γίνεται προσπάθεια εισαγωγής στη βάση χρήστη που υπάρχει ήδη.

### UserExistsException( string e)

Η πρώτη παράμετρος του exception είναι το email του χρήστη.

### virtual const char \* what ()

Επιστρέφει με μορφή string τα στοιχεία του exception.

## class AlreadyFriendsException

Exception για την περίπτωση που γίνει προσπάθεια να γίνει φίλος κάποιος που είναι ήδη φίλος.

### AlreadyFriendsException (User \*pu1, User \*pu2)

Η πρώτη παράμετρος του exception είναι ο χρήστης pu1 που προσπαθεί να κάνει φίλο τον χρήστη pu2.

### virtual const char \* what ()

Επιστρέφει με μορφή string τα στοιχεία του exception.

## class AlreadyLikeException

Exception για την περίπτωση που γίνει προσπάθεια για Like σε μήνυμα που έχει ήδη γίνει Like.

### AlreadyLikeException(User \*pu, Message \*pm)

Η πρώτη παράμετρος του exception είναι ο χρήστης και η δεύτερη το μήνυμα

### virtual const char \* what ()

Επιστρέφει με μορφή string τα στοιχεία του exception.

## class NoFriendsException

Exception για την περίπτωση που γίνει προσπάθεια ενέργειας που προορίζεται μόνο για φίλους (π.χ. αποστολή μηνύματος) σε κάποιον που δεν είναι φίλος.

### NoFriendsException (User \*pu1, User \*pu2)

Η πρώτη παράμετρος του exception είναι ο χρήστης pu1 που προσπαθεί να κάνει ενέργεια στον χρήστη pu2.

### virtual const char \* what ()

Επιστρέφει με μορφή string τα στοιχεία του exception.

## class UserNotExistsException

Exception για την περίπτωση που γίνεται προσπάθεια πρόσβασης σε χρήστη (χρησιμοποιώντας το email του) που δεν υπάρχει.

### UserNotExistsException( string e)

Η πρώτη παράμετρος του exception είναι το email του χρήστη.

### virtual const char \* what ()

Επιστρέφει με μορφή string τα στοιχεία του exception.

## class SameUserException

Exception για την περίπτωση που γίνει προσπάθεια να κάνει κάποιος φίλο τον εαυτό του.

### SameUserException( string e)

Η πρώτη παράμετρος του exception είναι ο χρήστης.

### virtual const char \* what ()

Επιστρέφει με μορφή string τα στοιχεία του exception.

# Γενικές συναρτήσεις

## void dotest()

Προσθέτει στο δίκτυο δοκιμαστικά δεδομένα.

## void domenu()

Εμφανίζει το μενού και εκτελεί κάθε επιλογή του χρήστη.

## int main()

Η κυρίως συνάρτηση του προγράμματος.

Αρχικά εκτελεί την dotest() για να προσθέσει στο δίκτυο τα δοκιμαστικά δεδομένα και να τα τυπώσει και στη συνέχεια την domenu για να τρέξει το μενού.

Παγιδεύει τις εξαιρέσεις που εγείρονται στο πρόγραμμα.