Το παρόν README, αφορά την Άσκηση 2, της Εργασίας 1.

class Student:

Η κλάση Student αναφέρεται στον μαθητή.

Το private section αποτελείται από ένα string name που ειναι το όνομά του, έναν integer Class που δείχνει την τάξη στην οποία ανήκει, ένα string gender που είναι το φύλο του και έναν integer behavior (boolean) που παίρνει την τιμή 0 ή 1 και δειχνει αν ο μαθητής έχει καλή (0) ή κακή (1) συμπεριφορά.

Στο public section βρίσκονται μόνο οι διάφορες συναρτήσεις που επεξεργάζονται τα παραπάνω members.

- *Student(string nam, int C, string gen)*
  Είναι ο constructor του Student που δέχεται ως ορίσματα το όνομα του, την τάξη που ανήκει και το φύλο του. Στο σώμα του, αρχικοποιούνται τα μέλη του Student βάσει των ορισμάτων και το member “behavior” παίρνει την τιμή “0” που σημαίνει ότι η συμπεριφορά του είναι καλή.
- *~Student()*
  Είναι ο destructor του Student. Στο σώμα του υπάρχει μέσα σε σχόλιο ένα cout για την επιβεβαίωση ότι δουλεύει σωστά.
- *void print()*
  Η συνάρτηση print εκτυπώνει το όνομα του μαθητή, την τάξη του και το φύλο του.
- *string getGEN()*
  Η συνάρτηση getGEN επιστρέφει στο όνομά της το member “gender” του μαθητή.
- *int getClass()*
  Η συνάρτηση getClass επιστρέφει στο όνομά της το member “Class” του μαθητή.
- *int getBEHAVE()*
- Η συνάρτηση getBEHAVE επιστρέφει στο όνομά της το member “behavior” του μαθητή.
- *void printNAME()*
  Η συνάρτηση printNAME εκτυπώνει το όνομα του μαθητή χωρίς αλλαγή γραμμής (endl).
- *int behave()*
  Η συνάρτηση ελέγχει αν ο μαθητής είναι ήσυχος (behavior=0) ή όχι (behavior=1). Αν είναι ήσυχος, τον κάνει άτακτο και επιστρέφει τον αριθμό “1” δηλώνοντας ότι έγινε η αλλαγή. Διαφορετικά επιστρέφει τον αριθμό “0” δηλώνοντας ότι είναι άτακτος.
- *void good()*
  Η συνάρτηση θέτει την τιμή του member “behavior” σε “0”.

class Order:

Η κλάση Order αναφέρεται στην ακολουθία των μαθητών.

Το private section αποτελείται από 2 pointers που δείχνουν σε Student και αποτελούν του 2 μαθητές της ακολουθίας, έναν integer “couple” που υποδηλώνει ποιο ζευγάρι της ακολουθίας είναι (πχ. couple=2 -> 2ο ζευγάρι) και έναν pointer “next” που δείχνει σε ένα αντικείμενο της κλάσης Order (ουσιαστικά το Order είναι node μιας συνδεδεμένης λίστας).

Στο public section βρίσκονται μόνο οι διάφορες συναρτήσεις που επεξεργάζονται τα παραπάνω members.

- *Order(int i=1)*
  Είναι ο constructor του Order που δέχεται ως μοναδικό όρισμα έναν integer, το οποίο αρχικοποιεί το member “couple”. Επίσης αρχικοποιεί τόσο τους 2 Student pointers όσο και τον self-pointer ως “NULL”.
- *~Order()*
  Είναι ο destructor του Order. Στο σώμα του υπάρχει μέσα σε σχόλιο ένα cout για την επιβεβαίωση ότι δουλεύει.
- *void enter(Student* st)*
  Η συνάρτηση enter έχει ως σκοπό να εισάγει έναν μαθητή στην ακολουθία. Το όρισμα της είναι ένας pointer στον Student που θέλουμε να εισάγουμε. Η συνάρτηση ψάχνει αν ο μαθητής μπορεί να μπει σε κάποια από τις υπάρχουσες θέσεις και αν όχι δημιουργεί ένα νέο αντικείμενο Order (Node), που το συνδέει με τα υπόλοιπα, και εισάγει τον μαθητή στην κατάλληλη θέση.
- *void print()*
  Η συνάρτηση print εκτυπώνει όλη την ακολουθία των μαθητών.
- *int solo()*
  Η συνάρτηση solo επιστρέφει στο όνομα της τον αριθμό “0” αν σε μια ακολουθία δεν έχει “μόνους” μαθητές (δηλαδή δεν έχουν ζευγάρι), “1” αν έχει αγόρι χωρίς ζευγάρι και “2” αν έχει κορίτσι χωρίς ζευγάρι.
- *Student* getLAST()*
  Η συνάρτηση getLAST σε συνδυασμό με την “solo()”, επιστρέφει στο όνομα της έναν pointer σε Student, που είναι ο μαθητής χωρίς κάποιο ζευγάρι.
- *int coups()*
  Η συνάρτηση coups επιστρέφει στο όνομα της το πλήθος των ζευγαριών που υπάρχουν στην ακολουθίας μιας τάξης.
- *void getINFO(int &ppl, int &coup, int &f)*
  Η συνάρτηση getINFO δέχεται 3 ορίσματα τα οποία και τροποποιεί και “επιστρέφει”. Αυτά είναι: ppl(~people), coup(~couples), f(~flag). Στο ppl επιστρέφει πόσα άτομα υπάρχουν στην ακολουθία μιας τάξης και στο coup τα ζευγάρια. Το f έχει ως default τιμή το “0”. Στην περίπτωση που coup>2, αν f=0 σημαίνει ότι δεν υπάρχουν περισσότερα από 2 συνεχόμενα ζευγάρια, ενώ αν f=1 σημαίνει το αντίθετο. Το “f” μας βοηθάει να αποφασίσουμε πως θα αντιμεταθέσουμε τους μαθητές.
- *Student* getBAD()*
  Η συνάρτηση getBAD επιστρέφει έναν pointer σε Student, που είναι ένας άτακτος μαθητής.
- *Student* swapRANDOM(Student* st)*
  Η συνάρτηση swapRANDOM δέχεται ως όρισμα έναν pointer σε Student. Θα επιλέξει τυχαία έναν μαθητή μέσα από την ακολουθία της τάξης που του υποδείξαμε κατα την κλήση, θα τον αφαιρέσει από τη θέση του και θα εισάγει τον μαθητή που περάσαμε με όρισμα. Τέλος θα επιστρέψει στο όνομα της τον μαθητή που αφαιρέθηκε.
- *int sumST()*
  Η συνάρτηση sumST επιστρέφει το πλήθος των μαθητών που βρίσκονται στην τάξη.

class Class:

Η κλάση Class αναφέρεται στην ακολουθία των τάξεων.

Το private section αποτελείται από έναν integer num που υποδηλώνει το πλήθος των τάξεων, έναν double pointer Order με όνομα “sequence” που ουσιαστικά είναι ένας μονοδιάστατος πίνακας που περιέχει δείκτες που δείχνουν στο 1ο στοιχείο της συνδεδεμένης λίστας που αναπαριστά την ακολουθία των μαθητών. Το μέγεθος του πίνακα αυτού, όπως και των υπόλοιπων, είναι τόσων θέσεων, όσο και το πλήθος των τάξεων (μιλώντας για το sequence[0] αναφερόμαστε στην ακολουθία μαθητών της πρώτης τάξης κ.ο.κ.). Επίσης περιέχει έναν μονοδιάστατο πίνακα coups που αποθηκεύει το πόσα ζευγάρια υπάρχουν σε κάθε ακολουθία και έναν πίνακα points που αποθηκεύονται οι πόντοι αταξίας κάθε τάξης.

Στο public section βρίσκονται μόνο οι διάφορες συναρτήσεις που επεξεργάζονται τα παραπάνω members.

- *Class(int K)*
  Είναι ο constructor του Class και δέχεται ως όρισμα έναν integer K που είναι το πλήθος των τάξεων και ανατίθεται στο member “num. Παράλληλα στο σώμα του γίνονται τα κατάλληλα memory allocations των πινάκων και παίρνουν κάποιες default τιμές.
- *~Class()*
  Είναι ο destructor του Class. Στο σώμα του υπάρχει μέσα σε σχόλιο ένα cout για την επιβεβαίωση ότι δουλεύει.
- *void enter(Student* st)*
  Η συνάρτηση enter δέχεται ως όρισμα έναν pointer σε Student που είναι ο μαθητής που θέλουμε να εισάγουμε. Στο σώμα καλείται η enter της κλάσης Order, για την ακολουθία της τάξης που ανήκει ο μαθητής.
- *void print()*
  Η συνάρτηση print, εκτυπώνει την ακολουθία των μαθητών για κάθε τάξη. Καλεί την συνάρτηση print της κλάσης Order για κάθε μια τάξη.
- *void normalization()*
  Η συνάρτηση normalization, “κανονικοποιεί” τις ακολουθίες των μαθητών. Με άλλα λόγια, ελέγχει αν υπάρχουν μαθητές, διαφορετικού φύλου, χωρίς ζευγάρι. Αν ναί, τους κάνει ζευγάρι μεταξύ τους.
- *void changes()*
  Η συνάρτηση changes καλείται για να γίνουν οι αντιμεταθέσεις μεταξύ των άτακτων μαθητών. Μέσα στο σώμα της, υπολογίζονται οι πόντοι αταξίας κάθε τάξης, ελέγχεται πως πρέπει να αντιμετωπίσουμε του μαθητές κάθε τάξης και γίνονται οι αντίστοιχες αλλαγές.
- *void infoCLASS(int Tquiet, int Tmessy)*
  Η συνάρτηση infoCLASS δέχεται ως ορίσματα 2 integers που είναι ποσοστά τοις εκατό (0<=T<=100, γίνεται κατάλληλος έλεγχος). Ελέγχει αν οι πόντοι, κάθε τάξης, είναι λιγότεροι ή περισσότεροι από τα αντίστοιχα ποσοστά των μαθητών και ανάλογα, εμφανίζει τα κατάλληλα μηνύματα για το πόσο ήσυχη ή άτακτη είναι κάθε τάξη.

ΛΕΙΤΟΥΡΓΙΑ MAIN ΚΑΙ ΡΟΗ ΠΡΟΓΡΑΜΜΑΤΟΣ.

Στη main δημιουργείται ο πίνακας που περιέχει pointers που δείχνουν σε Student. Αφού γίνει memory allocation, διαβάζω από ένα αρχείο (“students.txt) τα δεδομένα των μαθητών.
>>!SOS! Τα δεδομένα είναι σε μορφή Name-ClassNum-Gender (πχ Mesrine-2-Male που σημαίνει ότι το όνομα του είναι Mesrine, η τάξη του είναι η 2η και το φύλο του είναι Male(~αγόρι)).
Στη main διαβάζω γραμμή-γραμμή τα δεδομένα και κάθε γραμμή είναι ένα string. Θέτω ως delimiter το σύμβολο “-” για να μπορώ να ξεχωρίζω τα διάφορα δεδομένα μεταξύ τους. Σε ένα πιο δυναμικό περιβάλλον θα μπορούσα να χρησιμοποιώ cin, ζητώντας από το χρήστη να μου δίνει 1-1 τα στοιχεία των μαθητών.
Αφού διαβαστούν, δημιουργηθούν και αποθηκευτούν τα αντικείμενα, δημιουργείται ένα αντικείμενο Class, καλώντας τον αντίστοιχο constructor.
Στη συνέχεια, γίνεται η εισαγωγή των μαθητών, ένας-ένας. Μετά την εισαγωγή, ακολουθεί η εκτύπωση των ακολουθιών κάθε τάξης. Μετά, καλείται η normalization, για να κάνει ζευγάρια τους “solo” μαθητές, αν υπάρχουν, και ακολουθεί ξανά εκτύπωση για να δούμε τις αλλαγές.
Επίσης, rand() αρχικοποιείται δίνοντας ως seed το time(NULL). Σε ένα for loop, για όσες φορές μας υποδείξει ο χρήστης (L), η rand() θα “αποφασίζει” τυχαία κάθε φορά το πλήθος των μαθητών που θα είναι άτακτοι και θα επιλέγει τυχαία τους μαθητές. Αφού, αυτοί οι μαθητές επισημανθούν ως “άτακτοι” καλείται η change() η οποία θα κάνει τις κατάλληλες αντιμεταθέσεις και θα υπολογίσει τους πόντου αταξίας κάθε τάξης.

- Οι πόντοι υπολογίζονται ως εξης. Διατρέχουμε την ακολουθία μιας τάξης:
    - Αν τα συνολικά ΖΕΥΓΑΡΙΑ μαθητών είναι λιγότερα ή ίσα από 2, τότε οι πόντοι αταξίας είναι τόσοι, όσοι και οι άτακτοι μαθητές (points = MessyStudents).
    - Αν είναι περισσότερα από 2, τότε οι πόντοι είναι το πλήθος των άτακτων μαθητών, επί 2 (points = MessyStudents*2).
- Οι αντιμεταθέσεις γίνονται ως εξης:
    - Αν τα συνολικά ΖΕΥΓΑΡΙΑ μαθητών είναι λιγότερα ή ίσα από 2, τότε οι αντιμεταθέσεις γίνονται εντός της ακολουθίας της ίδιας τάξης. Με άλλα λόγια, παίρνω κάθε φορά τον 1ο άτακτο μαθητή που θα βρω, επιλέγω έναν τυχαίο μέσα από την ακολουθία, τους αντιμεταθέτω και θέτω και των 2 τη συμπεριφορα ως “μη-άτακτη” (αρα αν ήταν και οι 2 άτακτοι, πλέον δεν είναι κανείς).
    - Αν είναι περισσότερα από 2:
        - Αν τα ΖΕΥΓΑΡΙΑ είναι διάσπαρτα, τότε παίρνω τον μαθητή και τον αντιμεταθέτω με έναν τυχαίο της επόμενη τάξης. Με άλλα λόγια, παίρνω κάθε φορά τον 1ο άτακτο μαθητή που θα βρω, επιλέγω έναν τυχαίο από την ακολουθία της επόμενης τάξης, τους αντιμεταθέτω και θέτω και των 2 τη συμπεριφορα ως “μη-άτακτη”. !ΠΡΟΣΟΧΗ! ένας μαθητής που πήγε στην επόμενη τάξη εξαιτίας της αντιμετάθεσης, μπορεί να ξαναγυρίσει στην τάξη του κατά την αντιμετάθεση ενός συμμαθητή του, λόγω της τυχαιότητας.
        - Αν τα ΖΕΥΓΑΡΙΑ είναι συνεχόμενα, τότε παίρνω τον μαθητή και τον αντιμεταθέτω με έναν τυχαίο μιας τυχαίας άλλης τάξης. Με άλλα λόγια, παίρνω κάθε φορά τον 1ο άτακτο μαθητή που θα βρω, επιλέγω έναν τυχαίο από την ακολουθία μιας τυχαίας τάξης, τους αντιμεταθέτω και

θέτω και των 2 τη συμπεριφορα ως “μη-άτακτη”. !ΠΡΟΣΟΧΗ! μπορεί να συμβεί, ό,τι και στο πάνω bullet.

ΑΛΛΑ ΔΙΑΔΙΚΑΣΤΙΚΑ & ΣΧΟΛΙΑ

Υπάρχει ένα makefile το οποίο έχει τις εξης εντολές:

- make
    - Δημιουργεί το εκτελέσιμο
- make clean
    - Σβήνει το εκτελέσιμο

Επίσης υπάρχει το students.txt που είναι απαραίτητο για την εκτέλεση του προγράμματος, καθώς η main είναι βασισμένη σε αυτό (σε περίπτωση που επέλεγα ένα πιο δυναμικό περιβάλλον προφανώς η αντιμετώπιση μου θα ήταν διαφορετική, αλλά υπήρχε ελαστικότητα πάνω σε αυτό.)

*Κατά την εκτέλεση του προγράμματος αρκεί να γράψετε τα εξης:

- make
- ask2 4 2 40 80

(Λόγω του πως έχω δομήσει το πρόγραμμα, είναι απαραίτητο το 1ο όρισμα να είναι ο αριθμός “4” που υποδηλώνει το πλήθος των τάξεων, καθώς το .txt αρχείο, κατανέμει του μαθητές σε 4 τάξεις. Τα υπόλοιπα ορίσματα μπορείτε να τα βάλετε ό,τι θέλετε.)

**Επίσης, στη main το memory allocation του πίνακα που περιέχει τους μαθητές, γίνεται “καθοδηγούμενα”, καθώς γνωρίζω το πόσοι ακριβώς είναι οι μαθητές (36). Στην περίπτωση που δεν ήξερα, θα εκανα διαφορετική υλοποίηση (πχ κάποια λίστα ή μόλις διάβαζα έναν μαθητή θα τον έκανα αμέσως enter στην τάξη του).