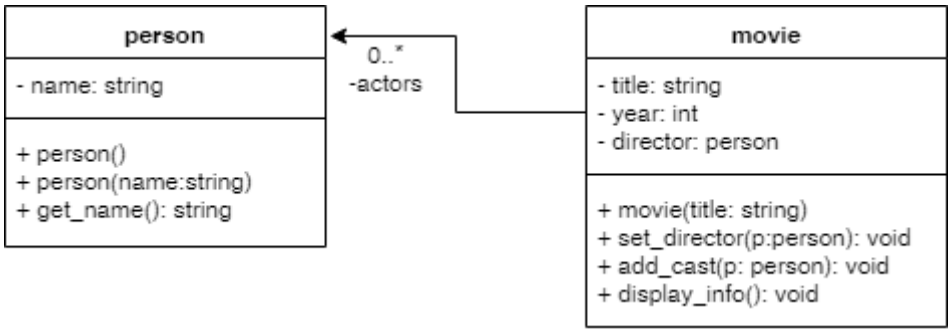
Αντικειμενοστραφής Προγραμματισμός με την C++

Τμήμα Πληροφορικής και Τηλεπικοινωνιών, Πανεπιστήμιο Ιωαννίνων

Ενδεικτικά θέματα για την τελική εξέταση (Γκόγκος Χρήστος)

Θέμα 1

Κατασκευάστε τις κλάσεις που δείχνει το ακόλουθο UML διάγραμμα κλάσεων (Σχήμα 1) και οι οποίες αναπαριστούν ταινίες (movie) που σκηνοθετούνται από ένα άτομο (person) και έχουν ως ηθοποιούς ένα σύνολο (std::vector) από άτομα. Στη συνέχεια δημιουργήστε 2 αντικείμενα ταινίες με τίτλους “Movie 1” και “Movie 2”, ορίστε ότι στη ταινία “Movie 1” είναι σκηνοθέτης ο “Director 1” και παίζουν οι ηθοποιοί “Actor 1” και “Actor 2” ενώ στη ταινία “Movie 2” είναι σκηνοθέτης ο “Director 2” και παίζουν οι ηθοποιοί “Actor 2” και “Actor 3”. Εμφανίστε καλώντας τη συνάρτηση μέλος display_info() της κλάσης movie τα στοιχεία της κάθε ταινίας, δηλαδή τίτλο, σκηνοθέτη και ηθοποιούς.



Σχήμα 1. Διάγραμμα κλάσεων (ταινίες, σκηνοθέτες και ηθοποιοί)

Θέμα 2

1. Δημιουργήστε μια κλάση rectangle (ορθογώνιο) με πεδία width (πλάτος) και height (ύψος).
2. Προσθέστε έναν κατασκευαστή που να δέχεται ως παραμέτρους τα δύο πεδία και να τα αρχικοποιεί.
3. Δημιουργήστε μια συνάρτηση area (εμβαδόν) που να επιστρέφει το εμβαδόν του ορθογωνίου.
4. Υπερφορτώστε τον τελεστή < έτσι ώστε να πραγματοποιεί σύγκριση αντικειμένων rectangle βάσει των εμβαδών τους.
5. Υπερφορτώστε τον τελεστή << έτσι ώστε να εκτυπώνει το πλάτος, το ύψος και το εμβαδόν του rectangle.
6. Δημιουργήστε ένα vector με 5 ορθογώνια.
7. Εμφανίστε το άθροισμα των εμβαδών από όλα τα ορθογώνια.
8. Ταξινομήστε όλα τα ορθογώνια με βάσει το εμβαδόν τους σε αύξουσα σειρά και εμφανίστε τα.
9. Ταξινομήστε όλα τα ορθογώνια με βάσει το εμβαδόν τους σε φθίνουσα σειρά και εμφανίστε τα.
10. Εμφανίστε το πλήθος των ορθογωνίων με εμβαδό άνω του μέσου όρου εμβαδών.

Θέμα 3

Α. Κατασκευάστε μια κλάση player (παίκτης) που:

- Να διαθέτει για κάθε παίκτη τα πεδία name (όνομα, std::string), dexterity (επιδεξιότητα, double) και wins (νίκες που έχει επιτύχει, std::vector<player>).
- Να έχει κατασκευαστή που να αρχικοποιεί έναν player δεχόμενο ως παραμέτρους ένα όνομα και μια τιμή επιδεξιότητας.
- Να έχει getter για το πεδίο dexterity.
- Να έχει μια συνάρτηση μέλος get_wins που να επιστρέφει το διάνυσμα wins.
- Να έχει μια συνάρτηση μέλος add_win που να προσθέτει στο διάνυσμα wins ένα άλλο παίκτη, με τη σημασία ότι ο τρέχων παίκτης έχει κερδίσει τον άλλο παίκτη σε μεταξύ τους αγώνα.

- Υπερφορτώστε τον τελεστή << έτσι ώστε να επιστρέφει για κάθε παίκτη έναν stream με το όνομα και την επιδεξιότητα του τρέχοντος παίκτη.

Β. Κατασκευάστε μια κλάση duel (μονομαχία) που:

- Να διαθέτει το πεδίο players (παίκτες, std::vector<player>).
- Να έχει έναν κατασκευαστή που να δέχεται ως παράμετρο έναν ακέραιο αριθμό **n** και να προσθέτει στο διάνυσμα των παικτών, **n** παίκτες με ονόματα της μορφής playerX όπου X θα είναι ένας ακέραιος από το 1 μέχρι το **n**. Για κάθε παίκτη η επιδεξιότητά του να είναι ένας τυχαίος πραγματικός αριθμός από το 0 μέχρι και το 100.
- Να έχει συνάρτηση μέλος get_players που να επιστρέφει το διάνυσμα των παικτών και συνάρτηση display_players που να εμφανίζει το διάνυσμα των παικτών.
- Να έχει μια συνάρτηση μέλος με όνομα next_round που να εκτελεί ένα γύρο μονομαχιών ανάμεσα σε ζεύγη παικτών ως εξής:
    - Να «ανακατεύει» τα περιεχόμενα του διανύσματος των παικτών.
    - Να εμφανίζει όλους τους παίκτες.
    - Να επιλέγει ζεύγη παικτών για να «μονομαχήσουν» με τον εξής τρόπο: ο πρώτος θα μονομαχεί με τον τελευταίο, ο δεύτερος με τον προτελευταίο κ.ο.κ.). Αν περισσεύει κάποιος παίκτης κατά το χωρισμό σε ζεύγη αντιπάλων τότε αυτόματα ο παίκτης που περισσεύει θα περνά στον επόμενο γύρο.
    - Να πραγματοποιεί αγώνα για κάθε ζεύγος παικτών και μόνο ο νικητής να διατηρείται στο διάνυσμα των παικτών. Σε κάθε αγώνα η πιθανότητα να κερδίσει ένας παίκτης είναι ανάλογη της επιδεξιότητάς του (π.χ. αν ένας παίκτης έχει επιδεξιότητα 70 και ένας άλλος 60, τότε ο πρώτος θα κερδίσει με πιθανότητα 70 / (70 + 60) = 70 / 130 = 53.85% και ο δεύτερος με πιθανότητα 100% - 53.85% = 46.15%).
    - Όταν ένας παίκτης Α κερδίζει έναν παίκτη Β τότε ο παίκτης Β να προστίθεται στις νίκες του παίκτη Α.

Γ. Κατασκευάστε μια main συνάρτηση που:

- Να δέχεται ως παράμετρο γραμμής εντολών το πλήθος των παικτών και να πραγματοποιεί ένα τουρνουά αγώνων. Αν ο χρήστης δεν εισάγει παράμετρο γραμμής εντολών τότε να πραγματοποιείται τουρνουά αγώνων με συμμετοχή 100 παικτών.
- Να εμφανίζει αρχικά τους παίκτες και στη συνέχεια να εκτελεί διαδοχικούς γύρους που εν τέλει θα καταλήγουν σε έναν μόνο παίκτη, που θα είναι ο νικητής.
- Να εμφανίζει τα στοιχεία του νικητή καθώς και τους αντιπάλους που κατάφερε να εξολοθρεύσει.

Παρατήρηση 1: για τη δημιουργία μιας τυχαίας πραγματικής τιμής στο διάστημα (0,1) μπορεί να χρησιμοποιηθεί ο ακόλουθος κώδικας:

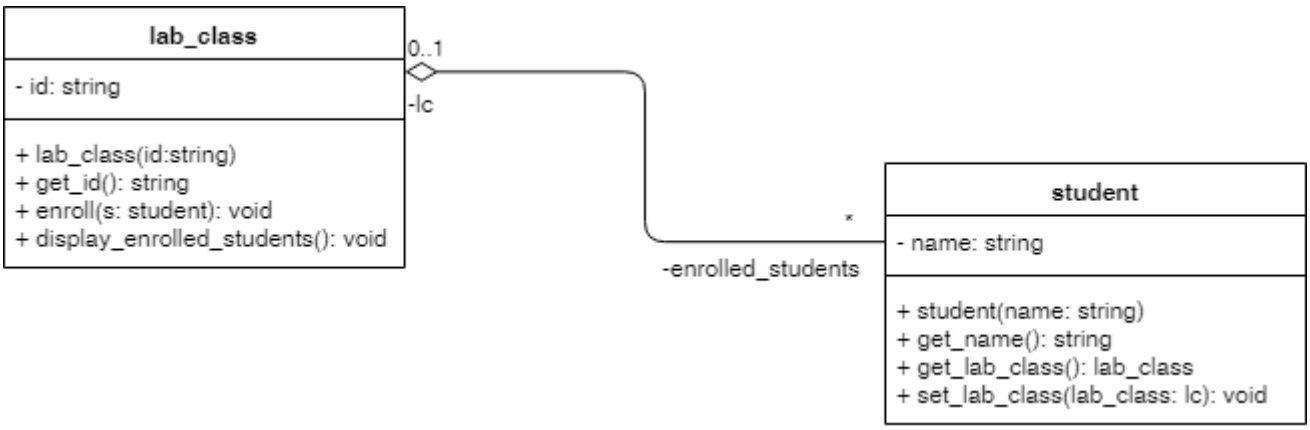
```
#include <random>
...
std::random_device rd;
std::mt19937 engine{rd()};
std::uniform_real_distribution<double> distribution(0.0, 1.0);
...
double r = distribution(engine);
...
```

Παρατήρηση 2: για το «ανακάτεμα» ενός διανύσματος **v** μπορεί να χρησιμοποιηθεί ο ακόλουθος κώδικας:

```
#include <vector>
#include <random>
...
std::random_device rd;
std::mt19937 engine{rd()};
...
std::shuffle(v.begin(), v.end(), engine);
...
```

Θέμα 4

Κατασκευάστε τις κλάσεις που αναπαρίστανται στο ακόλουθο UML διάγραμμα κλάσεων (Σχήμα 2). Η κλάση student αναπαριστά φοιτητές και η κλάση lab_class αναπαριστά τμήματα εργαστηρίου. Κάθε φοιτητής μπορεί να είναι εγγεγραμμένος σε ένα εργαστήριο το πολύ και κάθε εργαστήριο μπορεί να έχει πολλούς φοιτητές.



Σχήμα 2. Διάγραμμα κλάσεων (φοιτητές και τμήματα εργαστηρίου)

Υλοποιήστε τις συναρτήσεις μέλη των δύο κλάσεων λαμβάνοντας υπόψη στην υλοποίηση της συνάρτησης enroll (εγγραφή σε τμήμα) ότι σε περίπτωση που ένας φοιτητής είναι εγγεγραμμένος σε ένα τμήμα δεν θα πρέπει να επιτρέπεται η εγγραφή του σε άλλο.

Στη main() του προγράμματος:

- Δημιουργήστε πέντε αντικείμενα φοιτητές και φοιτήτριες με ονόματα kostas, petros, maria, christina και tasos και δύο αντικείμενα τμήματα εργαστηρίου με κωδικούς OOP1 και OOP2.
- Εγγράψτε τους 2 πρώτους φοιτητές στο τμήμα OOP1 και τους δύο τελευταίους στο τμήμα OOP2. Προσπαθήστε να εγγράψετε τον kostas στο εργαστήριο OOP2 έτσι ώστε να εμφανιστεί μήνυμα σφάλματος καθώς είναι ήδη εγγεγραμμένος στο τμήμα OOP1.
- Εμφανίστε λίστα εγγεγραμμένων φοιτητών για κάθε τμήμα.

Θέμα 5

Κατασκευάστε τις κλάσεις που αναπαρίστανται στο ακόλουθο UML διάγραμμα κλάσεων (Σχήμα 3).
Για την κλάση person:

- Ορίστε τα πεδία της ως protected.
- Ορίστε τη συνάρτηση info ως pure virtual.

Για την κλάση employee:

- Ορίστε τα πεδία της ως protected.
- Ορίστε τη συνάρτηση info ως virtual, η οποία θα επιστρέφει σε ένα λεκτικό όλα τα πεδία της κλάσης employee.

Για την κλάση professor:

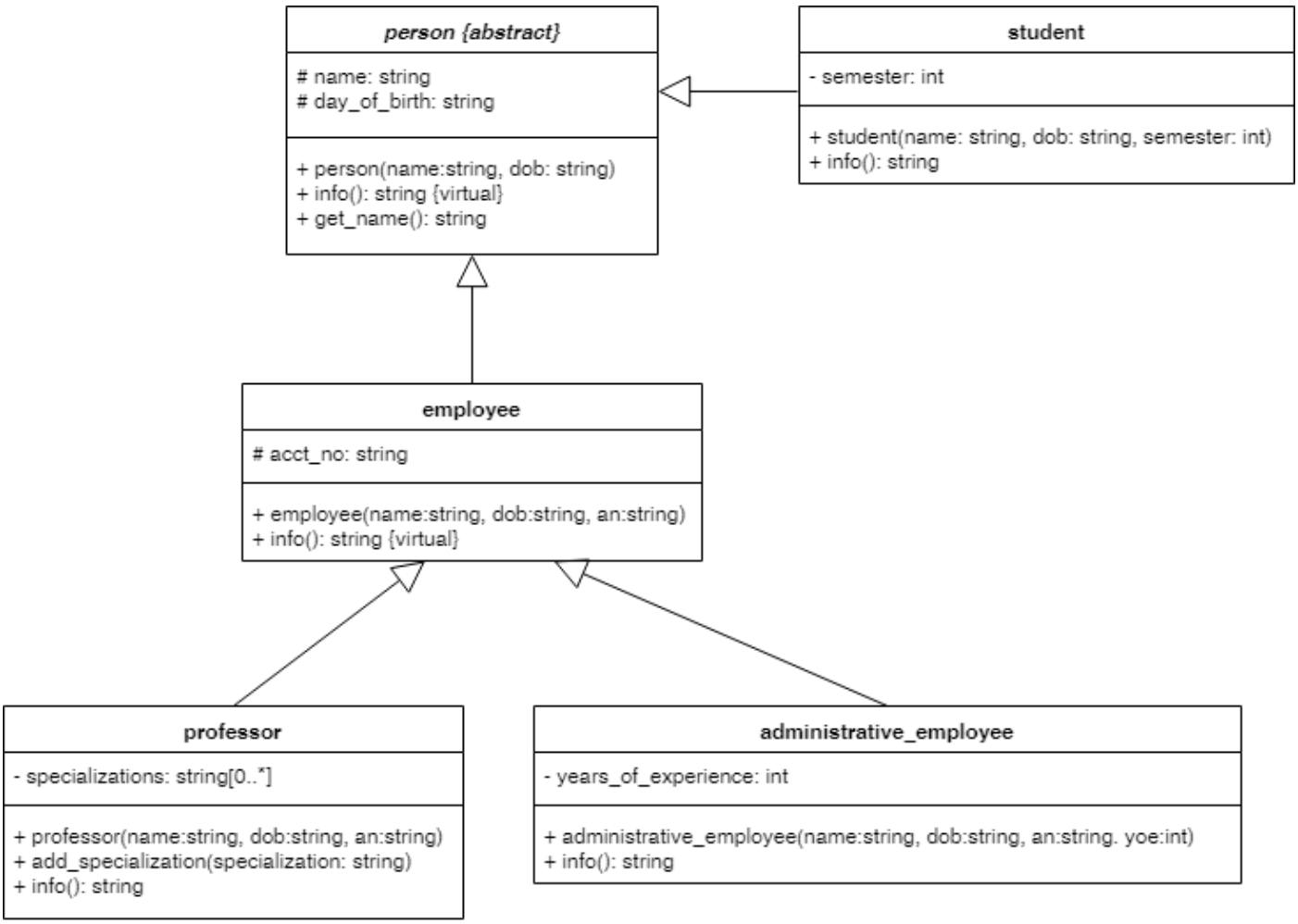
- Ορίστε το πεδίο specializations (εξειδικεύσεις) ως std::vector<std::string>.
- Ορίστε τη συνάρτηση info έτσι ώστε να επιστρέφει σε ένα λεκτικό όλα τα πεδία της κλάσης professor.
- Ορίστε τη συνάρτηση add_specialization έτσι ώστε να προσθέτει ένα λεκτικό στο διάνυσμα specializations.

Για την κλάση administrative_employee:

- Ορίστε τη συνάρτηση info έτσι ώστε να επιστρέφει σε ένα λεκτικό όλα τα πεδία της κλάσης administrative_employee.

Για την κλάση student:

- Ορίστε τη συνάρτηση info έτσι ώστε να επιστρέφει σε ένα λεκτικό όλα τα πεδία της κλάσης student.



Σχήμα 3. Διάγραμμα κλάσεων (πολυμορφική ιεραρχία)

Στη main() του προγράμματος:

- Δημιουργήστε ένα αντικείμενο employee με όνομα “Nikos”, ημερομηνία γέννησης “1/1/1980” και αριθμό λογαριασμού “0001”.
- Δημιουργήστε ένα αντικείμενο professor με όνομα “Maria”, ημερομηνία γέννησης “1/6/1965” και αριθμό λογαριασμού “0002”.
- Δημιουργήστε ένα αντικείμενο administrative_employee με όνομα “Kostas”, ημερομηνία γέννησης “2/4/1985” και αριθμό λογαριασμού “0003”.
- Δημιουργήστε ένα αντικείμενο student με όνομα “Georgia”, ημερομηνία γέννησης “1/3/2003” που βρίσκεται στο 2^{ο} εξάμηνο.
- Εμφανίστε τα αποτελέσματα που επιστρέφει η συνάρτηση info όταν κληθεί για καθένα από τα παραπάνω αντικείμενα.
- Τοποθετήστε τα αντικείμενα σε ένα std::vector<person*>, ταξινομήστε το vector σε αύξουσα σειρά ονομάτων και εμφανίστε τα αποτελέσματα.

Οι απαντήσεις βρίσκονται στο https://github.com/chgogos/oop