Organizarea laboratoarelor

0. Ce îmi doresc să câștigăm din acest laborator

Vom ști să folosim Programarea Orientată pe Obiecte în C++, probabil cel mai dificil limbaj în care putem ști asta :)

Ne vom dezvolta gândirea în detaliu. Tot C++ ne ajută cu asta, mai ales pe partea de POO.

Vom scrie mai eficient și mai rapid cod.

Vom putea explica clar lucruri considerate dificile la interviuri.

Alte abilități tehnice: vom putea folosi colecțiile și algoritmii de bază din STL, vom putea gestiona memoria alocată dinamic mai ușor cu shared_ptr si unique_ptr

Voi ce vă mai doriți?

1. Laboratoarele

Vom coda strict în C++. Practic materia este una stufoasă în care ne axăm pe detalii.

Limbajul nu este cel mai modern, dar vom folosi feature-uri moderne (și semi-moderne ;).) ale lui, precum STL și smart pointers, care ne vor simplifica viața. Măcar noi să facem asta.

Vom folosi editor modern, ca să fim eficienți și acum, și în orice veți coda în viitor.

Nu vom reinventa roata, dar vom înțelege cum a fost construită :)).

2. **Proiecte**

Vom avea 3 proiecte și un Colocviu (în ultima săptămână).

Cele trei proiecte valoreaza 25% din nota finala, iar Colocviul 25% din nota finală.

De menționat: Puteți obține notele 11 si 12 la laborator.
Nota voastră finală de laborator (proiecte + colocviu) va fi folosită în a calcula media cu examenul.

Pe principiul: Aveți o zi mai proastă la examen? Chiar un semestru întreg nu se poate întâmpla asta.

Proiectele vor avea termen limită până la final de săptămânile 4, 8, 12.
Săptămânile 1-4 vor destul de calme, 5-8 vor fi mai avansate iar 9-11 sunt proba decisivă.
În săptămânile 12-14 deja ne vom pregăti pentru colocviu.

3. **Evaluare este individuală,**
   creșterea va fi mai bună impreună. Sunteți o grupă sau un grup de oameni cu potențial?
   Dacă un coleg drag de-al vostru obține un bonus când codează într-un anumit fel, de ce să nu obținem și noi?

Primul Laborator

Ne facem încălzirea

https://education.github.com/pack <- Putem sa aplicam aici pentru a primi in maxim 15 zile licenta pt editorul CLion.
Il vom descarca deja de aici https://www.jetbrains.com/clion/download/ deoarece are trial de 30 zile, dar trebuie creat un *cont pe GitHub* (este de preferat sa folositi adresa voastra ***@s.unibuc.ro***)

Să codăm din nou în C++. Pe lângă a folosi C++, să ne concentrăm pe câteva exerciții cu două scopuri: 1 vom lucra cu date compuse (an, luna și ziua au sens doar împreună), 2 vom atinge și exerciții de interviu.

1. Se citesc de la tastatură 6 variabile, denumite an1, luna1, ziua1, an2, luna2, ziua2, care vor reprezenta două date calendaristice.
   Se cere să afișați:
   a) Care dintre cele două date calendaristice este mai mare?
   b) Care dintre cele două date calendaristice este mai apropiată de ziua de astăzi?

E.g.

```
2 10 2021
3 9 2021
a) Data 2.10.2021 este mai mare
b) Data 2.10.2021 este mai apropiată
```

```
7 2 2022
25 2 2022
a) Data 25.2.2022 este mai mare
b) Data 7.2.2022 este mai apropiată
```

2. Se citește de la tastatură un șir de n numere naturale.
   Se cere să
   a) Afișați toate fracțiile **unice** supraunitare care pot fi create folosind două elemente distincte dintre cele citite,
   b) Afișați fracțiile în ordine crescătoare (Bonus).

   E.g.

n=**4**
2 8 10 4
8/2 10/2 4/2 10/8 10/4
Afișate crescător:
10/8 4/2 10/4 8/2 10/2

3. Se citește de la tastatură un număr natural n și un șir cu 2*n numere naturale (adică două șiruri de câte n elemente fiecare).
   Se cere
   a) Reordonarea primelor n elemente și a ultimelor n elemente astfel încât suma **`s = v1[0] * v2[0] + v1[1] * v2[1] + … + v1[n-1] * v2[n-1]`** să aibă o valoare cât mai mare.
   Care credeți că este modalitate corectă de a ordona cele două șiruri?
   Cum ați proceda dacă ne dorim ca **s** să fie cât mai mic?

4. Interesant și simplu. Se cere descompunerea în fracții egiptene a unei fracții citite de la tastatură (două numere naturale **a** și **b**).
   Algoritmul este simplu. Fracțiile egiptene sunt acestea.
   a) Ce idei aveți?

   Dacă aveți nevoie de hint-uri:
   Pe scurt, vom repeta procedeul: Găsim cea mai mare fracție egipteana (de forma 1/x) mai mică decât a/b. Daca 1/x este chiar a/b, am încheiat. Altfel, recalculam a și b (scădem din fractia **a/b** valoarea fracției **1/x**).

   E.g.

$$\frac{13}{12} = \frac{1}{2} + \frac{1}{3} + \frac{1}{4}$$

Bonus 1

5. Se citește de la tastatură data de început și data de final a n evenimente. Similar cu exercițiul 1, vom avea data=an,luna,zi. Ne putem imagina că avem o listă de festivale, tabere, vacanțe posibile și ne dorim să participăm la cât mai multe, dar unele dintre acestea se suprapun.

   Ce modalitate știți de a crea o listă cu cât mai multe evenimente care nu se suprapun?

Hint: Haideți să folosim mai multe funcții pentru a împărți exercițiul în bucăți ușor de rezolvat:

Vom avea nevoie de o funcție care citește de la tastatură un eveniment.

Avem nevoie să aflăm dacă o dată este mai mare decât o altă dată.
Vom avea și o funcție care afișează un eveniment.
Si in plus, o funcție care sortează toate evenimentele citite. După ce principiu sunt sortate?

Întrebare de autoevaluare (în grup ;))

Puteți refolosi codul creat de voi pentru a rezolva aceiași cerință, dar dacă la eveniment adăugăm o denumire (char nume[100]), dar si ora de inceput? Cât la sută din codul vostru (sau care porțiuni de cod) ar trebui modificate?
Va fi important: codul bun poate fi refolosit și modificat ușor :).

Ne dezvoltăm prin Feedback. Pentru a vă spune ce punctaj bonus obțineți (ăsta este motivul pentru care cerințele b) și 5 și altele sunt mai dificile) puteți trimite soluții pe GDrive sau la adresa stefan.deaconu@s.unibuc.ro dacă Google nu vă permite să încărcați, deoarece deja le-ați umplut serverele cu cod de calitate ;).

Bonus 2

6. Back to divide et impera in C++: https://www.infoarena.ro/probleme-de-acoperire-1 puteți face problema 1 pentru a exersa o funcție recursiva. Consider bonus daca faceți 2 probleme de aici.