Konstrukcijske vježbe pravila

1. Minimum koji studenti MORAJU imati je implementirani CRUID. Create-Read-Update-Insert-Delete i tu se misli na kontekst sadržaja koji se zapisuje u datoteku koja može biti tekstualna ili binarna. Dalje samo nadograđujete svoj program!
2. Studenti su dužni dolaziti na konstrukcijske vježbe, ako imaju pitanja i susreću se s određenim izazovima, nastavnik je dostupan u terminu konstrukcijskih vježbi da pomogne savjetom ili smjernicom u rješavanju problema.
3. Na konstrukcijskim vježbama nastavnik prati studente i dodjeljuje im bodove za njihovu aktivnost. Moguće je ostvariti maksimalno 4 boda po konstrukcijskoj vježbi prema predloženim implementacijama različitih koncepata.
4. Student treba koristiti materijale s predavanja, laboratorijskih vježbi, a tek onda s interneta i druge stručne literature.
5. Realno je kako student neće imati dovoljno vremena na svim konstrukcijskim vježbama za završetak svog projekta, stoga je očekivano kako student mora odvojiti većinu svoga vremena kod kuće za rješavanje svog projekta. To je definirano i ECTS bodovima kojih ima ukupno 8, prvo predavanje.
6. Student bi trebao riješiti projektni zadatak prije kraja konstrukcijskih vježbi. Maksimalne bodove iz KV dobiva samo ako su implementirani svi obavezni koncepti. Ako nema sve obavezne koncepte implementirane, morat će ih implementirati, dolaziti na konstrukcijske vježbe i raditi kod kuće.
7. Program je gotov kad se implementiraju svi obavezni koncepti, kada je program testiran i ne ruši se niti u jednoj točki izvršavanja. Mora pokazati program nastavniku na konstrukcijskim vježbama.
8. Postoje ostale specifičnosti koje nisu navedene u prijašnjim točkama i koje će biti praćene od strane nastavnika, kao na primjer neuredno pisan kôd, ne konvencionalna upotreba imenovanja varijabli, konstanti i sl.
9. Ako student tvrdi da je uspio ranije napraviti cijeli program, nastavnik koji vodi svoju grupu odlučit će je li program uistinu kompletno dovršen.
10. Ako student ne stigne dovršiti svoj program tijekom konstrukcijskih vježbi, dovršit će samostalno, testirati i tek kada je program stabilan (ne ruši se niti u jednoj točki i uvijete), može prijaviti pismeni ispit.
11. Ako se tijekom usmenog ispita program sruši, usmeni odmah završava ili ako nisu implementirani svi obavezni koncepti.
12. Kada student odluči izaći na usmeni ispit obavezan je prijaviti pismeni ispit koji prvi dolazi, a ne unaprijed prijaviti tri roka. Ispitivač će objaviti je li usmeni u terminu pismenog ispita ili koji dan kasnije.
13. Programski kôd, kao i sve ostale datoteke, kao što su tekstualne ili binarne datoteke, moraju biti postavljene na platformu kao što su github, gitlab, bitbucket ili nešto četvrto. Ovim načinom se pojednostavljuje rad na konstrukcijskim vježbama, nastavak rada kod kuće i tako u krug.

Obavezni koncepti

1. Odabir konkretnih primitivnih tipova podataka za rad s cjelobrojnim i realnim konstantama.
2. Odabir konkretnih složenih tipova podataka za rad sa specifičnim objektima.
3. Primjena typedef sa strukturama i unijama, po potrebi s enum tipovima podataka tamo gdje treba.
4. Imenovanje identifikatora (varijabli, konstanti, polja, funkcija, pokazivača…) – upotreba camelCase, PascalCase i snake_case konzistento kroz cijeli projekt.
5. Primjena ključne riječi static za globalne i lokalne varijable.
6. Organizacija izvornog kôda.
7. Primjena extern ključne riječi za globalne varijable.
8. Ako su funkcije jednostavne koristiti makro funkcije ili inline funkcije.
9. Izbornik/podizbornici.
10. Generalno upotreba pokazivača tamo gdje su potrebni.
11. Generalno upotreba struktura i funkcija.
12. Zaštita parametara kod svih funkcija.
13. Koristiti statički zauzeta polja gdje su potrebna, nikako ne koristiti VLA polja.
14. Koristiti dinamičko zauzimanje memorije za bilo koji tip podatka, osobito za složene tipove podataka.
15. Koristiti funkcije malloc(), calloc(), realloc(), free() – neku od njih, ako ne i sve.
16. Sigurno brisanje memorije koja je dinamički zauzeta, anuliranje memorijskog prostora, provjera pokazivača kako se ne bi dogodila pogreška double free() i anuliranje svih pokazivača koji su bili usmjereni na memorijski prostor koji se dinamički zauzeo.
17. Datoteke, koristiti tekstualnu ili binarnu, provjera pokazivača i zatvaranje datoteke.
18. Koristiti funkcije fseek(), ftell(), rewind(), ovisno o potrebi – neku od njih ako ne sve.
19. Koristiti funkcije remove(), rename(), po potrebi implementirati funkciju za kopiranje datoteka.
20. Upravljati s pogreškama, errno, perror(), strerror(), feof(), ferror() – neku od njih ako ne sve.
21. Sortiranje – preporuka koristiti ugrađenu qsort() funkciju.
22. Pretraživanje – preporuka koristiti ugrađenu bsearch() funkciju.
23. Rekurzije je najlakše koristiti primjenom rekurzivnih algoritama sortiranja kao što su quick sort, merge sort, insert sort ili pretraživanja kao što je binary search.
24. Pokazivače na funkcije je najlakše koristiti upotrebom funkcije qsort() ili bsearch() iz standardne biblioteke.

Dopunski koncepti:

1. Jednostruko ili dvostruko povezani popis.

Savjeti:

1. Minimalna upotreba globalnih varijabli.

2. Funkcije moraju biti kratke, konkretne. Trebaju raditi samo jedan posao i to dobro. Riješiti problem dijeljenjem na više manjih rješenja i povezivanje dobivanje glavnog rješenja pomoću funkcija.
3. Nikako ne koristiti VLA (engl. Variable lenght arrays) – koristiti dinamičko zauzimanje memorije i obavezno potpuno rukovanje memorijom (provjera pokazivača, oslobađanje memorije).