Kreiran je jednostavan pokazni program kako bi trebao izgledati projekt studenata kada završe KV iz kolegija Programiranje 2SR.

Program nije savršen, nije jedini način kako se može napraviti, ima još mjesta za reduciranje ponavljajućeg kôda i dodatnog iskorištavanja već napisanih funkcija.

Izbornici se mogu ljepše napisati, može se brisati stari sadržaj izbornika unutar konzolne aplikacije, kao i pauziranje između pojedinih opcija.

Program sam pokušao napraviti s ChatGPT, neke se stvari uspijevaju modelirati, ali i dalje je potreban veliki zahvat programera za ispravljanje grešaka i kreiranje kvalitetnijeg i svrsishodnijeg programa.

Program ispunjava sadržajno više od minimuma, a 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.

Ovaj program na početku pita korisnika za unos imena datoteke s proširenjem s kojom će raditi. Zatim se ulazi u izbornik koji nudi nekoliko opcija:

Binary File Operations Menu:
0. Exit
1. Insert Student
2. Update Student
3. Delete Student
4. Find Student
5. Read Students
6. Delete Binary File
7. Rename Binary File
8. Sort Students Menu
9. Search Students Menu
Enter your choice:

Izbornik ne mora biti uopće ovako koncipiran, to je čisto pokazno. Možda su neke opcije združene jedna unutar druge kao podizbornik ili se neke funkcionalnosti pozivaju jedna iza druge jednom od odabranih opcija iz izbornika.

Testirao sam program kada ne postoji datoteka u svakoj točki i koliko sam uspio istražiti, program se ne ruši. Studentima se isto tako program ne smije rušiti niti u jednoj točki izvršavanja.

Ako je pogreška prevelika, program se treba mirno vratiti u glavni izbornik ili prekinuti rad.

Koje koncepte studenti moraju koristiti za dovršavanje svog projekta?

- Strukture
- Pokazivači na strukture
- Enumeracije
- Typedef

- Organizacija izvornog kôda u više datoteka, primjena static i extern ključne riječi s funkcijama i globalnim/lokalnim varijablama
- Funkcije – prijenos parametara po vrijednosti ili memorijskoj adresi, primjena const ključne riječi za zastitu parametara. Moja preporuka je prijenos parametara po memorijskoj adresi.
- Koristiti izbornik i podizbornik
- Dinamičko i statičko zauzimanje polja, apsolutno nikako ne koristiti VLA polja.
- Prilikom upotrebe dinamičkog zauzimanja memorije apsolutno paziti na provjeru pokazivača kao i oslobađanje memorije nakon upotrebe.
- Korištenje datoteka, binarne ili tekstualne
- Prilikom otvaranje datoteke paziti na provjeru pokazivača, kao i zatvaranje toka prema datoteci.
- Obavezna upotreba sortiranja i pretraživanja, bilo kroz funkcije standardne biblioteke ili preuzetih algoritama čije su implementacije implementirane u kôd projekta
- Rekurzije se najlakše mogu koristiti kroz implementiranje algoritama sortiranja i pretraživanja
- Pokazivači na funkcije se mogu najlakše implementirati kroz primjenu funkcije qsort() ili bsearch() iz standardne biblioteke

Više o pravilima konstrukcijskim vježbama i konceptima možete pročitati u dokumentu „Konstrukcijske vježbe pravila“ koja se nalaze na Merlin stranicama kolegija.

Bodovanje na KV ide tako da se 1 bod dobije što su došli, a ostala tri po bod za svaku implementirani obavezni koncept koji može biti bilo koji od 24 koje sam naveo po KV. To je ukupno 4 boda po KV, a ima ukupno 4 KVa.

Vratimo se nazad na primjer projektnog zadatka.

Od datoteka zaglavlja imamo 4 datoteke, file.h, file_types.h, menu.h, student.h. Svaka od tih datoteka zaglavlja ima svoju datoteku s proširenjem .c osim datoteke zaglavlja file_types.h.

U datoteci zaglavlja file.h nalaze se deklaracije funkcija koje rade isključivo s binarnom datotekom na disku. Pretprocesorska naredba/direktiva #pragma once štiti od višestrukog umetanja datoteke zaglavlja unutar datoteke izvornog kôda s proširenjem .c.

Datoteka zaglavlja file_types.h sadrži definicije složenih tipova podataka koji su typedef. Ova datoteka nema pragma direktivu već kombinaciju pretprocesorskih naredbi koje rade istu stvar, a to je da štite od višestrukog umetanja datoteke zaglavlja unutar datoteke izvornog kôda. Konstanta je napisana velikim slovima i umjesto .h mora ići donja crta _ koja se ujedno koristi i za rastavljanje riječi ako se naziv sastoji od više riječi. Ključna riječ typedef nam služi samo da damo kraće i intuitivnije ime tipu podatka i ne mijenja stari naziv, ali ako se već typedefa, onda molim lijepa koristiti novi naziv! Izbacio sam ime strukture u enuma pasu anonimni, stoga se može koristiti samo novi naziv dobiven typedef.

Ključnom riječi struct se kreira složeni tip podatka strukture, slično kao klasa u OOP jezicima samo bez metoda koje operiraju nad atributima klase/ strukture i svi članovi u strukturi su public!

Ključnom riječi enum kreira se brojani tip podatka kojem je svaki član cjelobrojna konstanta. Ako se drugačije ne navede, prvi član enuma nosi vrijednost 0, a svaki sljedeći za jedna veći od prethodnog. Ako se prvi član ili bilo koji unutar liste konstanti postavi na neki cijeli broj različit od 0, sljedeći članovi su za jedan veći od onih koji su prethodili.

Datoteka menu.h je dosadna ima ima malo sadržaja .

Datoteka student.h je bogatija sadržajem :P .

Obratiti pozornost kako parametri unutar deklaracija imaju postavljene const ključne riječi, pa da samo ponovimo, const ispred tipa podatka čini sadržaj varijable nepromjenjiv, a const nakon zvjezdice čini pokazivač konstantnim pokazivačem koji nije moguće preusmjeriti sa memorijske adrese koja se u njemu nalazi.

U datoteci main.c nalazi se main() funkcija koja će biti dosta mala jer smo program rasporedili u funkcije. U ovu datoteku su uključena samo ona zaglavlja koja su od interesa, tj. čije funkcije koristimo u toj datoteci izvornog kôda. Obratite pozornost na uključivane korisnički kreirana zaglavlja, ona se pišu unutar dvostrukih navodnika, dok sistemska unutar znakova manje i veće!

Kad primijetite korištenje getchar() funkcije u projektu na različitim mjestima prije dohvaćanje stringa, to je zato što je vjerojatno iz prethodnog unosa sadržaja s tipkovnice u input spremniku ostao enter '\n', pa ga getchar() pojede i omogući čisti unos novog stringa s tipkovnice.

Također ste primijetili kako se u nekim datoteka izvornog kôda koristi pretprocesorksa naredba #define _CRT_SECURE_NO_WARNINGS, samo iz razloga što se Microsoftov prevoditelj buni kada se radi sa standardnim IO funkcijama i traži C11 standard funkcije koje su upitne jer Microsoft forsira ipak svoje C11 funkcije. Ova pretprocesorska naredba se MORA pisati na samom vrhu datoteke izvornog kôda inače nula bodova od potiskivanja poruka od prevoditelja.

Datoteka menu.c sadrži izbornik koji za određene brojeve koje predstavljaju opcije u izborniku koristi konstante iz enumeracije radi bolje čitljivosti kôda, a ne da stoje „Magic numbers“ i ne zna se za što se koriste.

Datoteka file.c funkcija createBinaryFile() kreira datoteku ako ne postoji, ako postoji nikome ništa, neće je prepisati. Svaki puta javlja poruku u ovisnosti kako je stanje s datotekom na disku.

Funkcija deleteBinaryFile() traži od korisnika potvrdu je li siguran da se obriše datoteka s diska, ako je potvrda sa stringom „yes“ tada se datoteka briše. Koristi se funkcija strupr() koja pretvara sva slova stringa u velika, ima i funkcija za obrnuto, ali loše je što je to POSIX funkcija i ne radi na prevoditelju VC++ od Microsoft, ali ima alternativa koja počinje s znakom podcrtano _strupr().

Funkcija renameBinaryFile() slično kao i brisanje samo što služi za preimenovanje.

Na kraju nam ostaje datoteka student.c u kojoj ima puno funkcija koje rade sa studentima.

Funkcija insertStudent() dodaje studenta na kraju datoteke.

Funkcija findStudent() služi za pronalazak studenta po kriteriju ID-a, kada ga nađe ispiše sve informacije o njemu i vrati pronađeni ID, ako nema takvog studenta vrati -1.

Funkcija updateStudent() služi za ažuriranje nekog studenta unutar datoteke. Služi se funkcijom findStudent() kako bi se pronašao određeni student, ako ga ima pronađemo ga, izmijenimo sve informacije, pomaknemo se za jedno mjesto u nazad jer se nakon njegova čitanja automatski pomakne na sljedećeg i onda se prepiše sadržaj unutar datoteke.

Funkcija deleteStudent() koristi se isto funkcijom findStudent() kako bi potvrdila postojanje studenta, zatim se kreira nova datoteka, u nju se upišu svi studenti osim ovog kojeg želimo obrisati. Obriše se stara datoteka, a nova se preimenuje u naziv stare sa sadržajem studenata osim onog kojeg smo htjeli obrisati. Znači ne postoji mehanizam za direktno brisanje iz datoteke već moramo ovako nešto raditi ili pomaći sve zapisa za sizeof(student_t) u nazad i pregaziti na taj način studenta unutar datoteke.

Funkcija readStudent() je generička funkcija za ispis sadržaja svih studenata.

Funkcija printStudent() je static funkcija koja živi i vidljiva je samo unutar datoteke student.c. To je pomoćna funkcija za funkciju sortStudentsMenu(). Njoj se preda polje studenata koje ona ispiše.

Funkcije compareByName...() i compare ByGrade...() su kriterijske funkcije koje se predaju funkcijama qsort() i bsearch() iz standardne biblioteke, odnosno predaju se mem. adrese tih funkcija. Kriterijske funkcije trebaju vratiti jednu od tri vrijednosti, -1, 0 ili 1!

Funkcija sortStudentsMenu() je podizbornik koji korisnika pita na koji način želi sortirati studente unutar memorije računala kada pročita sav sadržaj iz datoteke u RAM. Ono što je interesantno je da funkcija fseek(file, 0, SEEK_END) radi s binarnom datotekom na Windowsima pod prevoditeljem VC++. Inače, konstanta SEEK_END dobro ne radi s binarnim datotekama jer nije dobro definirano njezino ponašanje na različitim OS. Sadržaj iz datoteke se pročita u HRPU (engl. Heap) i onda se korisnika pita kako hoće da se sadržaj sortira, uzlazno ili silazno po imenu ili ocjeni. Koriste se enumeracijske konstante unutar podizbornika.

Funkcija searchStudentsMenu() isto je podizbornik u kojemu se korisnika pita hoće li se pretraga raditi po kriteriju imena ili ocjene. Koristi se ugrađeni bsearch() koji traži da polje bude prvo sortirano uzlazno, a to se postiže ugrađenom funkcijom qsort(). Koriste se enumeracijske konstante unutar podizbornika.