# Konvecije i principi za modularno i strukturirano pisanje koda za projekat iz predmeta PPuTViOS1

## Opšta pravila

Kod koji se piše treba da bude uniformno formatiran, iskomentarisan i modularan. Format koda nije bitan dok god je uniforman kroz ceo kod i praktičan (kod gde ne postoji indentacije jeste uniforman, ali nije praktičan). Komentari u kodu treba da se nalaze na svim mestima gde funkcionalnost kod nije očigledna iz samog koda, tj. potrebno je duže vreme da bi se funkcionalnost koda shvatila. Komentari ne treba da postoje "čisto zbog radi toga" već da imaju svoju svrhu.

## Struktuiranje izvornih fajlova

Svaki izvorni fajl, **nezavisno od toga da li je .h ili .c**, treba da uključuje (makro `#include`) **samo one .h fajlove koje direktno koristi**. Da ne bi došlo do višestrukog uključivanja fajlova, svaki .h fajl treba da bude uokviren sledećim makroima:

```
#ifndef _UNIKATAN_MAKRO_ZA_OVAJ_H_FAJL_
#define _UNIKATAN_MAKRO_ZA_OVAJ_H_FAJL_

//ovde ide stvaran sadržaj .h fajla

#endif
```

Takođe treba napomenuti da se korišćenjem ovog pristupa eliminiše potreba da se npr. u primerA.h fajlu uključi primerB.h fajl, ukoliko je on neophodan za kompajliranje primerA.c fajla.

## Sadržaj modula

Jedan modul se najčešće sastoji od jednog para .h i .c fajlova. Iako to nije nužno slucaj, sasvim je dovoljno za ovaj projekat. Jedan modul treba da pruža skup srodnih funkcionalnosti (npr. modul za parsiranje tabela, modul za rukovanje ulaznim događajima sa daljinskog upravljača, itd.). Jedan modul sme da se oslanja samo na na svoje funkcije i tipove, sistemske funkcije i tipove, i funkcije i tipove modula na koje se oslanja. Zavisnost između modula ne može biti dvosmerna, tj. ako modul M1 zavisi od modula M2, onda modul M2 ne može zavisiti od modula M1.

.h fajl za jedan modul predstavlja njegov **API**. U njemu treba da se nalaze svi tipovi (`typedef` definicije, definicije struktura, itd.) i makroi - `#define` (samo ukoliko su neophodni za komunikaciju sa modulom ili za definiciju tipova - npr. makro koji definiše broj elemenata u nizu

koji je polje u strukturi) koju su neophodni za komunikaciju sa tim modulom. Pored tipova u .h fajlu treba da se nalaze i prototipovi funkcija koje taj modul pruža. Pomoćne funkcije koje se koriste unutar samog modula se ne definišu u .h fajlu. Na primer, ukoliko modulu pruža funkcionalnost parsiranja PAT tabele funkcijom sa sledećim prototipom:

```
void parsePAT(uint8_t* buffer, PATTable* patTable);
```

i prototip funkcije i definicija tipa `PATTable` trebaju biti definisani u .h fajlu.

Unutar samog .h fajla, prvo se pisu definicije makroa, zatim definicije tipova i na kraju prototipi funkcija.

Implementacija funkcija **API-ja** modula treba da se nalazi u .c fajlu. Na početku .c fajla treba da se nalze i sve pomoćne funkcije koje se koriste od strane funkcija koje predstavljaju API. U .c fajlu takođe treba definisati i lokalne promenljive za modul. Pomoćne funkcije treba da budu označene sa ključnom reči `static` kako ne bi mogle da se koriste nigde van samog fajla, tj. kako se njihovi simboli ne bi nalazili u objektnom fajlu (u iskompajliranoj biblioteci).

**Globalne promenljive** (promenljive definisane u jednom fajlu, a referencirane iz drugog pomoću ključne reči `extern`) **se strogo ne preporučuju**. Postoji jako malo razloga da se takve promenljive koriste, a ne postoje za potrebe ovog projekta.

Sve u funkcije u jednom modulu (isto pravilo može da važi i za sve module) treba da imaju uniformna pravila za vraćanje povratne vrednosti. Na primer, može se uzeti pravilo da se sve povratne vrednosti vraćaju preko pokazivača, a da sama povratna vrednost svake funkcije u stvari označava kod greške. **Strukture kao argumenti** se uglavnom **ne šalju po vrednosti**, vec po referenci zbog zauzeća memorije na steku. Isto važi i za vraćanje strukture kao vrednosti (treba ih vraćati putem pokazivača.

Značenje povratne vrednosti, značenje argumenata funkcija, svrhu funkcije kao i druge značajne stvari vezane za API modula, treba uredno staviti u komentar iznad prototipa funkcija u .h fajlu. Preporučuje se da komentari budu napisani po sintaksi za generisanje dokumentacije altom **doxygen**, ali nije neophodno.

Nazivi tipova, promenljivih i funkcija takođe treba da prate uniformna pravila (kamilja notacija, mađarska notacija, notacija u linux kernelu) koja će važiti minimalno u jednom modulu, a preporučljivo je za ceo projekat. Sam princip nazivanja se ostavlja kao izbor studentu, samo dok je uniforman i praktičan.

## Predlog podele modula i njihovih funkcionalnosti

Ovde će biti opisan predlog podele modula u projektu i skup funkcionalnosti koji svaki modul treba da podrži.

Projekat se može podeliti na sledeće module (funkcionalnosti):

- modul za rukovanje događajima sa daljinskog upravljača
- modul za čuvanje stanja grafičkih elemenata i njihovo iscrtavanje
- modul za rukovanje televizijskim funkcionalnostima koji se dalje može podeliti na više modula
    - modul za rukovanje tuner-om, player-om i listom kanala
    - modul za parsiranje tabela

### Modul za rukovanje događajima sa daljinskog upravljača

Ovaj modul treba da podržava registraciju callback funkcije koja će se prozvati svaki put kada se desi novi koristan događaj na daljinskom upravljaču (za implementaciju callback-a neophodno je korišćenje pokazivača na funkcije - ako želite pitajte asistenta). S obzirom na to da je čekanje na događaje blokirajuća operacija, samo čekanje treba izdvojiti u posebnu nit. Radi lakše implementacije, preporučuju se i funkcije za inicijalizaciju (npr. otvaranje uređaja i pokretanje niti koja će da čeka na događaje) i deinicijalizaciju (zatvaranje uređaja i čekanje na završetak niti) modula.

### Modul za čuvanje stanja grafičkih elemenata i njihovo iscrtavanje

Ovaj modul treba da podržava postvaljanje parametara za prikaz (npr. teksta u info banner-u) grafike na ekranu. Sam ekran se može osvežavati na promenu (kada god se nešto u prikazu promeni), na zahtev (postoji funkcija koja iscrtava sve podatke na ekran) ili periodično (ekran se osvežava svakih *n* milisekundi). Prilikom implementacije ovog modula treba imati u vidu da svaki put treba obrisati sadržaj bafera i iscrtati **sve** ponovo, jer prethodno iscrtane ekran (eng. *frame*)ostaje u baferu dok se ne očisti.

### Modul za rukovanje televizijskim funkcionalnostima

#### *Modul za rukovanje tuner-om, player-om i listom kanala*

Ovaj modul treba da podržava funkcionalnosti kao sto su inicijalno skeniranje kanala i formiranje liste kanala, prebacivanje na kanal, zaustavljanje reprodukcije kanala, dobavljanje podataka iz transportnog toka (eng. *transport stream*), itd. Kako je parsiranje tabela iz transportnog toka poseban problem, prirodno je da se ovaj modul oslanja na drugi modul koji će se baviti samo tim problemom.

#### *Modul za parsiranje tabela*

Ovaj modul je pomoćni i na raspolaganje korisniku (programeru korisniku) stavlja funkcije koje parsiraju različite tabele iz transportnog toka. Kao ulazni argument primaju niz bajtova koji u sebi sadrže tabelu za parsiranje i popunjavaju strukturu koja sadrži podatke iz tabele.