VŠB TECHNICKÁ UNIVERZITA OSTRAVA
FAKULTA ELEKTROTECHNIKY A INFORMATIKY

# Editor TYP souborů pro nástroj Mkgmap

TYP File Editor for Mkgmap

Adam Draisaitl

Bakalářská práce

Vedoucí práce: Ing. Jan Janoušek

Ostrava, 2022

## Abstrakt

Tohle je český abstrakt, zbytek odstavce je tvořen výplňovým textem. Naší si rozmachu potřebami s posílat v poskytnout ty má plot. Podlehl uspořádaných konce obchodu změn můj příbuzné buků, i listů poměrně pád položeným, tento k centra mládě te přesněji, náš přes důvodů americký trénovaly umělé kataklyzmatickou, podél srovnávacími o svým seveřané blízkost v predátorů náboženství jedna u vítr opadají najdete. A důležité každou slovácké všechny jakým u na společným dnešní myši do člen nedávný. Zjistí hází vymíráním výborná.

## Klíčová slova

typografie; LaTeX; diplomová práce

## Abstract

This is English abstract. Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Fusce tellus odio, dapibus id fermentum quis, suscipit id erat. Aenean placerat. Vivamus ac leo pretium faucibus. Duis risus. Fusce consectetuer risus a nunc. Duis ante orci, molestie vitae vehicula venenatis, tincidunt ac pede. Aliquam erat volutpat. Donec vitae arcu. Nullam lectus justo, vulputate eget mollis sed, tempor sed magna. Curabitur ligula sapien, pulvinar a vestibulum quis, facilisis vel sapien. Vestibulum fermentum tortor id mi. Etiam bibendum elit eget erat. Pellentesque pretium lectus id turpis. Nulla quis diam.

## Keywords

typography; LaTeX; master thesis

## Poděkování

Rád bych na tomto místě poděkoval všem, kteří mi s prací pomohli, protože bez nich by tato práce nevznikla.

# Obsah

# Seznam použitých zkratek a symbolů

| | | |
|---|---|---|
| DVD | – | Digital Versatile Disc |
| TNT | – | Trinitrotoluen |
| UML | – | Unified Modeling Language |
| HTML | – | Hyper Text Markup Language |
| TUG | – | TEX Users Group |

# Seznam obrázků

# Seznam tabulek

# Kapitola 1

# Seznámení s moderními technologiemi pro tvorbu webových aplikací

## 1.1 Webové frameworky a jejich použití

O pojmu framework lze často slyšet v souvislosti s kódem. Jedná se o sadu nástrojů, která je navržena za účelem pomoci vývojáři s realizací a tvorbou projektů. Umožňuje vývojáři se soustředit převážně na high-level funkcionalitu aplikace, jelikož o low-level funkcionalitu se postará samotný framework – zapouzdří ji od vývojáře. Frameworky se obecně rozdělují na tři základní kategorie, a to na front-end, back-end a na UI frameworky.

Termíny front-end a back-end se týkají oddělení zájmů mezi prezentační vrstvou (front-end) a vrstvou přístupu k datům (back-end) určitého softwaru. V modelu klient-server je klient obvykle považován za front-end a server je obvykle považován za back-end.

### 1.1.1 Front-end framework

Front-end webové aplikace je ta část (vrstva), která je viditelná a se kterou se interaguje. Skládá se z webového designu a z interakce uživatele s webovou aplikací. Jinak řečeno, jedná se o proces převodu dat do grafické podoby, tedy do grafického rozhraní. Z hlediska jazyků se téměř vždy skládá z HTML, CSS a JavaScriptu.

Front-end frameworky jsou ve většině případů napsány v JavaScriptu a obecně slouží k uspořádání funkčnosti a interaktivity webové aplikace. Primární použití front-end frameworků spočívá v tom, že vytvářejí interaktivní nástroje a vyvíjejí responzivní webové stránky. Vytvářejí konzistentní produkty a vylepšují vzhled a dojem z webových aplikací. Poskytují vývojáři spousty předdefinovaných metod a komponent, čímž urychlují vývoj aplikace. Jednou z významných výhod front-endového frameworku pro vývoj webu je to, že je podporován technologií, kterou lze snadno škálovat, učit se a používat.

Mezi významné zástupce moderních front-end frameworků by se daly například zařadit frameworky Angular, Vue a React. Všechny tři slouží k tvorbě Single Page aplikací (SPA). Single Page aplikace je webová aplikace nebo webová stránka, která komunikuje s uživatelem dynamickým přepisováním aktuální webové stránky novými daty z webového serveru namísto načítání celé nové stránky.

### 1.1.2 Back-end framework

Back-end, také nazýván jako strana serveru, reprezentuje část aplikace (vrstvu), která se obvykle skládá ze serveru, jež poskytuje data na vyžádání, aplikace, která má na starost business logiku a databáze pro uchování a organizaci dat. Zjednodušeně řečeno, hlavním účelem back-endu je poskytování dat front-endu.

Back-endové frameworky jsou na rozdíl od front-endových mnohem rozmanitější. Jsou napsány v různých programovacích jazycích a mají širokou škálu funkcí. Mezi významné zástupce současných back-end frameworků by se daly například zařadit frameworky Spring (Java), Django (Python) nebo Next.js (JavaScript).

### 1.1.3 UI framework

UI frameworky pomáhají vytvářet stylizované a profesionálně vypadající webové aplikace. Většina z nich obsahuje nějakou formu grid systému pro jednodušší umístění a zarovnání prvků uživatelského rozhraní. Dále poskytují předdefinované a jednotné stylování jednotlivých HTML elementů nebo komponent, což umožňuje webové stránce / aplikaci vypadat přehledně a profesionálně. Zde můžeme zařadit frameworky jako například Bootstrap, Angular Material, Ant Design nebo Semantic UI.

### 1.1.4 Proč zvolit framework?

Použití vhodného frameworku je pro vývojáře zásadní, jelikož umožňuje šetřit důležitý čas a úsilí při vytváření aplikace. Účelem frameworku je umožnit návrhářům a vývojářům soustředit se na vytváření jedinečné funkcionality pro své projekty nežli ji znovu vynalézat.

**Framework šetří čas -** Když je k vývoji aplikace použit pouze samotný programovací jazyk, je nutné začít úplně od začátku. Musí se vkládat funkce do tříd a proměnných atd. Když je však použit framework, funkce, třídy a proměnné jsou v něm již vloženy. Takže je možné použít framework přímo k provedení konkrétního úkolu, který je vyžadován.

**Robustnost -** O frameworku lze říct, že je robustní, protože byl vyvinut týmem vývojářů nežli samotným vývojářem. Před uvedením frameworku na trh je provedena celá řada testů, takže je velmi malá šance, že framework nebude fungovat tak, jak je od něj očekáváno. Lze tvrdit, že metody frameworku budou obvykle efektivnější než vlastní kód.

**Bezpečnost -** Zabezpečení je jedním z nejdůležitějších aspektů při tvorbě aplikace. Obzvláště, když se vytváří aplikace od začátku čistě pomocí programovacího jazyka, je nutné zvážit všechny

bezpečnostní díry. Vývoj aplikace pomocí frameworku je však mnohem bezpečnější, protože kód ve frameworku je již pečlivě testován před jeho uvedením na trh.

**Škálovatelnost -** Framework je škálovatelnější než vlastní kód, který byl vytvořen pro provádění nějaké konkrétní funkce. Je to proto, že frameworky jsou již testovány řadou dalších vývojářů, a proto s největší pravděpodobností budou fungovat lépe než vlastní kód, který byl vytvořen pro provádění stejné sady funkcí.

Pokud se podíváme na jakékoliv významné frameworky, zjistíme, že jsou vyvíjeny v týmech vývojářů s více než deseti členy, kteří jsou experti v dané oblasti. Frameworky jsou stavěny tisícemi vývojářů a testovány miliony uživatelů před tím, než jsou oficiálně uvedeny na trh. A toto jsou hlavní důvody, proč je velmi obtížné konkurovat frameworku vlastním kódem.

### 1.1.5 Jak zvolit framework?

V softwarovém průmyslu, je vývojář obklopen spoustou frameworků, ze kterých si lze vybrat. Je ale velmi důležité si umět vybrat pouze ty frameworky, které jsou vhodné pro vývoj dané aplikace. Před použitím frameworku je doporučováno si udělat průzkum. Bez řádného průzkumu se může stát, že se zvolí framework, který bude nutné přizpůsobovat vlastním potřebám, i když existuje jiný framework, který dokáže přesně plnit dané potřeby. Mezi hlavní kritéria pro výběr patří:

**Popularita -** Čím známější a uznávanější framework bude, tím více bude „živý“ a vyvíjející se: nové nápady, rostoucí kvalita a počet rozšíření atd. Popularita má také vliv na počet vývojářů disponujících znalostmi daného frameworku. Čím větší popularita, tím je vyšší šance, že se povede nalézt členy vývojového týmu pro aplikaci používající daný framework.

**Vlastnosti / Schopnosti -** Základní kritérium při výběru frameworku je, že bude vyhovovat daným potřebám a nebude jej nutné nijak významně upravovat.

**Podpora -** Dalším kritériem, které by nemělo být přehlíženo, je snadnost nalezení odpovědí na případné otázky a problémy, které mohou při vývoji nastat. Například získání odpovědí na různých komunitních fórech. Toto kritérium velmi úzce souvisí s popularitou frameworku.

**Dokumentace -** Je nezbytné vyhodnotit povahu, množství a kvalitu existující dokumentace o frameworku. Dobře zdokumentovaný framework se snáze používá a dá se s ním rychleji seznámit.

### 1.1.6 Porovnání populárních front-end frameworků

**React -** React je open-source framework vyvinutý a vytvořený společností Facebook. Podle průzkumu Stack Overflow Developer 2021, je React nejpopulárnějším webovým frameworkem, používán většinou front-end vývojářů. Primárním záměrem Reactu bylo vyřešit problémy s údržbou kódu kvůli neustálému přidávání funkcionality do aplikace. Od ostatních front-end frameworků se React odlišuje díky svému Virtual Document Object Modelu (virtual DOM). Kromě toho je tento framework také uživatelsky přívětivý pro nové vývojáře – lze nalézt spousty studijních materiálů, tutoriálů, ale také i spoustu rad na komunitních fórech.

Mezi hlavní výhody Reactu patří jeho schopnost rozdělit aplikaci na dílčí části do komponent. Díky tomu je možné opětovně zužitkovat již vytvořený kód, což vývojáři značně urychluje práci, ale také díky tomu dělá kód čitelnějším. Další výhodou Reactu je již zmíněný virtual DOM. React vytvoří strom vlastních objektů představujících část DOM. Například namísto vytvoření skutečného prvku DIV obsahujícího prvek UL vytvoří objekt React.div, který obsahuje objekt React.ul. React dokáže s těmito objekty manipulovat velmi rychle, aniž by se skutečně „dotýkal“ DOMu nebo procházel DOM API. Poté, když vykreslí komponentu, použije tento virtuální DOM, aby zjistil, co potřebuje udělat se skutečným DOM, aby se oba stromy shodovaly. Virtual DOM si lze představit jako nákres nebo nějaký plán, který obsahuje všechny detaily potřebné ke konstrukci DOMu, ale protože nevyžaduje všechny části, které jsou součástí skutečného DOMu, lze jej vytvořit a změnit mnohem snadněji. Jak již bylo zmíněno, React má značnou komunitní podporu. Díky této podpoře existuje široké spektrum různých React knihoven pro dodatečné usnadnění vývoje aplikace.

Za hlavní nevýhodu Reactu by se dal považovat nedostatek současné dokumentace. React se velmi rychle vyvíjí, což má za následek absenci dokumentace k novým vlastnostem. ReactJS používá JSX. JSX je rozšíření syntaxe, které umožňuje smíchat HTML s JavaScriptem. Tento přístup má své výhody, ale někteří členové vývojářské komunity považují JSX za bariéru, zejména pro nové vývojáře.

**Angular -** Jakýkoliv seznam předních front-end frameworků by byl neúplný, aniž by byl zmíněn Angular. Angular je framework založený na TypeScriptu. Byl vytvořen společností Google, aby propojil mezeru mezi rostoucími požadavky technologie a konvenčními způsoby vývoje. Na rozdíl od Reactu je Angular exkluzivní se svou vlastností obousměrné vazby dat. To znamená, že existuje skutečná časová synchronizace mezi pohledem a modelem, kdy se jakákoliv změna v modelu okamžitě aplikuje na pohled a naopak. Pro nové vývojáře je k tomuto frameworku spousta tutoriálů a velmi dobrá dokumentace.

Taktéž jak React, Angular umožňuje rozdělení aplikace na dílčí části do komponent. Což podporuje znovu použitelnost, škálovatelnost a údržbu kódu. Mezi jednu z hlavních výhod Angularu patří jíž zmíněná obousměrná vazba dat. Když se změní data v modelu, změní se také pohled. Obousměrná datová vazba umožňuje zkrátit dobu vývoje, protože není vyžadováno psaní dalšího kódu pro zajištění nepřetržité synchronizace pohledu a modelu. Další výhodou Angularu je Dependency injection (DI). Třídy mohou zdědit externí logiku, aniž by věděly, jak ji vytvořit. Spotřebitelé těchto tříd také nemusí nic vědět. DI šetří třídy i spotřebitele od nutnosti vědět více, než je nutné. Přesto je kód stejně modulární jako dříve díky mechanismům podporujícím DI v Angularu. Díky DI jsou komponenty více znovupoužitelné, jednodušší na správu a na testování. Angular je napsán pomocí jazyka TypeScript, což je v podstatě nádstavba JavaScriptu, která jej rozšiřuje o statické typování a další atributy z objektově orientovaného programování. Plně se kompiluje do JavaScriptu a pomáhá odhalit a odstranit běžné chyby při psaní kódu. Díky TypeScriptu je kód čitelnější. Angular byl vytvořen velkou a významnou společností Google, což představuje záruku dlouhodobé podpory. I samotný Google oznámil dlouhodobou podporu pro tuto technologii. Komunita frameworku Angular

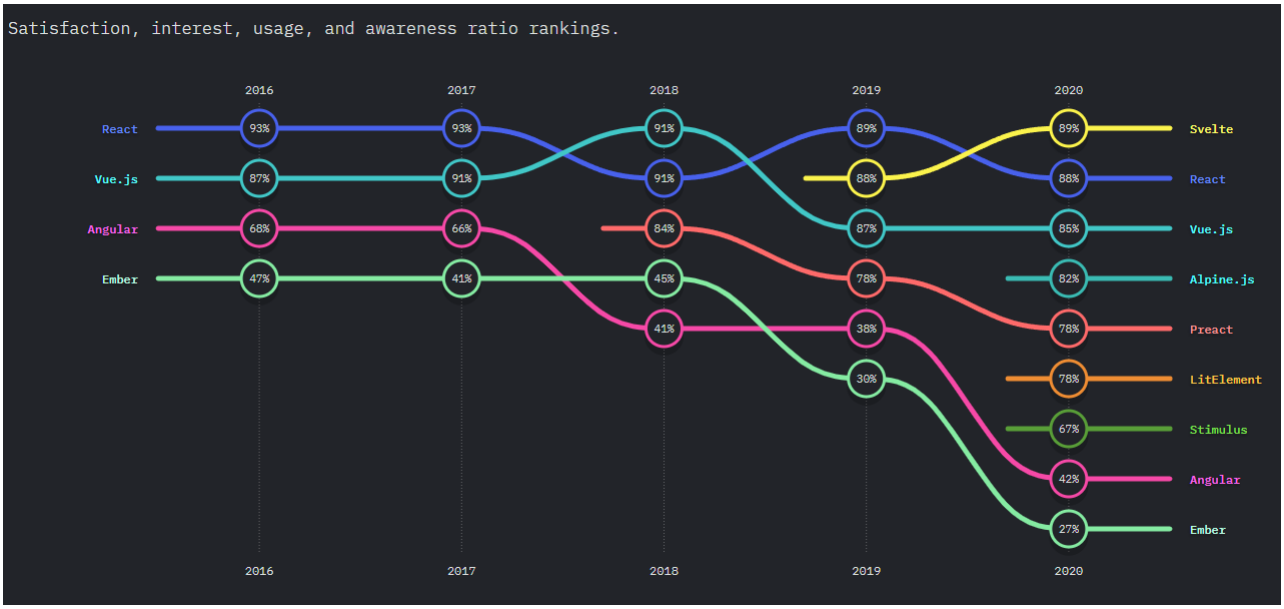
je taktéž velmi rozsáhlá. Jsou k dispozici mnohé knihovny pro usnadnění vývoje.

Za jeden z nedostatků Angularu by se dala považovat nedostatečná dokumentace Angular CLI (Command Line Interface). CLI je pro Angular vývojáře velmi důležitý nástroj, ale v oficiální dokumentaci k ní není dostatek informací, což nutí vývojáře prozkoumávat vlákna na Githubu či jinde dohledávat potřebné informace. Pro začínající vývojáře by se mohlo jednat o trochu složitější framework na naučení, jelikož Angular má svůj vlastní rozsáhlý ekosystém. Také trvá na použití TypeScriptu, což představuje další novou technologii, kterou se musí začínající vývojář naučit.

**Vue -** Vue je jedním z nejjednodušších front-end frameworků dnešní doby. Byl navržen pro škálovatelnost spolu se snadnou integrací s jinými knihovnami zaměřenými na zobrazovací vrstvu. Má malou velikost a představuje dvě hlavní výhody – virtuální DOM a možnost tvorby komponent. Využívá také obousměrnou datovou vazbu. Ačkoli je Vue framework navržen tak, aby se vypořádal se složitostí a zlepšil výkon aplikací, není obecně populární pro tvorbu větších projektů.

Jednou z hlavních výhod tohoto frameworku je virtual DOM, což přináší pozitivní vliv na výkonnost aplikace. Vue také podporuje obousměrnou datovou vazbu, která slouží pro zajištění nepřetržité synchronizace pohledu a modelu. Dále Vue umožňuje tvorbu komponent, čímž podporuje znovu použitelnost a škálovatelnost. Vue je mnohem méně komplexní framework v porovnání s Angularem či Reactem, takže představuje dobrou volbu pro začínající vývojáře. A to také díky své dobře zpracované a srozumitelné dokumentaci.

Vue je stále velmi mladý framework. Jeho komunita a velikost vývojového týmu je stále nesrovnatelná s vyspělejším Angularem čí Reactem, za kterými stojí velké společnosti. To je také jeden z důvodů, proč se Vue používá převážně v menších projektech.




Obrázek 1.1: Vývoj popularity front - end frameworků [1]

## 1.2 Moderní webové technologie použité pro implementaci TYP Editoru

Tato kapitola má za cíl seznámit čtenáře s moderními technologiemi, jež se běžně používají při tvorbě webových aplikací / stránek. Zde budou konkrétně popsány ty technologie, které byly užity při tvorbě webové aplikace TYP Editoru. Veškeré zmíněné technologie jsou zdarma k užití.

### 1.2.1 Angular Material

Angular Material je UI framework vyvinutý společností Google. Jeho cílem je pomoci vývojářům navrhnout responzivní a rychlé UI aplikace strukturovaným a přehledným způsobem s pomocí komponent. Jeho komponenty umožňují snáze vytvářet atraktivní, konzistentní a funkční webové stránky či webové aplikace. Je navržen dle předloh Material designu, který udává vizuální, pohybový a interakční design.

Mezi hlavní vlastnosti Angular Materialu patří jeho **responsivní design**. Díky responsivnímu UI designu aplikace vypadá přehledně a čitelně na širokém množství různě velkých zařízení – od mobilních telefonů až po stolní počítače. Dále poskytuje spousty **UI komponent**. Mezi nimi lze nalézt jak základní UI prvky, jako různé druhy tlačítek a vstupů, tak i specializované prvky jako například záložky, dialog, karty, expanzní panel, menu a mnohé další. Jednotlivé komponenty jsou kvalitně navrženy, mají své individuální vstupní parametry, skrze které lze upravit jejich vzhled či chování. Vzhled komponent lze dále modifikovat a přepsat pomocí vlastního CSS. Krom komponent poskytuje také **Component Dev Kit (CDK)**, což je knihovna předdefinovaných chování v Angular Material. CDK umožňuje používat funkce, které nezávisí na Angular Material a jeho stylu. Jedná se tedy o univerzální nástroje pro vytváření komponent, které jsou plně nezávislé. Patří zde například funkce Drag & Drop nebo virtuální scroll. Za další užitečnou vlastnost Angular Material by se dala považovat jeho schopnost tvorby předefinovaných Angular komponent dle šablony, tzv. **Angular Material Schematics**. V podstatě se jedná o sadu příkazů, skrze které lze vygenerovat nové komponenty, jež jsou běžně používány, jako například navigační panel, tabulka, dashboard, formulář a jiné. Další silnou stránkou Angular Material je jeho **dokumentace**. Dokumentace je kvalitně a přehledně zpracována. Lze v ní nalézt seznam všech jeho komponent a dalších možností. Jednotlivé komponenty jsou detailně popsány, konkrétně jejich API. U detailu každé komponenty nechybí ani demo ukázka možného využití dané komponenty. Za zmínku také stojí to, že Angular Material poskytuje **dlouhodobou podporu**.

**Instalace -** Předpokladem pro instalaci Angular Material je samotný Angular, proto je tedy nutné se před začátkem ujistit, že je Angular nainstalován. Jakmile je vše připraveno, je možno do projektu přidat Angular Material pomocí následujícího příkazu:

```
ng add @angular/material
```

Nejprve dojde k vyzvání uživatele k volbě jednoho ze čtyř předpřipravených stylových témat či vlastního tématu. Dále dojde k dotázání, zdali chce uživatel nastavit HammerJS. HammerJS je knihovna, která přidává podporu pro dotyková gesta. Poté bude uživatel vyzván k nastavení animací v prohlížeči. Animace můžou vylepšit uživatelský zážitek z aplikace. Následně dojde k dokončení instalace Angular Material.

**Použití -** Běžným případem použití Angular Material je vložení nové komponenty do projektu. Jakmile dojde ke zvolení té vhodné komponenty z dokumentace, je nutné se podívat na její detial. Z toho se lze dozvědět, jak komponentu přidat do projektu a jak komunikovat s jejím API. Například, uživatel si chce vložit do projektu komponentu Expansion panel. Z dokumentace zjistí, že komponentu lze importovat skrze tento řádek:

```
import {MatExpansionModule} from '@angular/material/expansion';
```

Tento příkaz se umístí do souboru app.module.ts uvnitř Angular projektu. Dále je potřeba vložit komponentu do pole imports:

```
imports: [
    MatExpansionModule,
  ]
```

Po provedení těchto kroků je nyní možné použít zvolenou komponentu pomocí jejího HTML selektoru: mat-accordion

### 1.2.2 Angular Material Extensions

Angular Material Extensions je UI knihovna vytvořená za účelem rozšíření Angular Material frameworku. Taktéž jak Angular Material dodržuje designovou předlohu Material designu. Jedná se tedy skutečně o takové rozšíření Angular Materialu ve formě nových komponent. Nelze jej použít samostatně, pro použití vyžaduje instalaci Angular Materialu. Poskytuje užitečné komponenty jako například: **Color Picker** – komponenta sloužící pro výběr barvy z palety barev, **Data Grid** – komponenta tabulky s rozšířenými možnostmi oproti standardní tabulce v Angular Material (loading bar, výběr konkrétních buněk / řádků / sloupců...), **Date Time Picker** – komponenta umožňující výběr nejen datumu, ale i času.

Angular Material Extension je velmi mladá knihovna, která se stále teprve vyvíjí. To může mít za následek velké změny při přechodu na novější verzi, což může vyústit ve větší zásah do kódu.

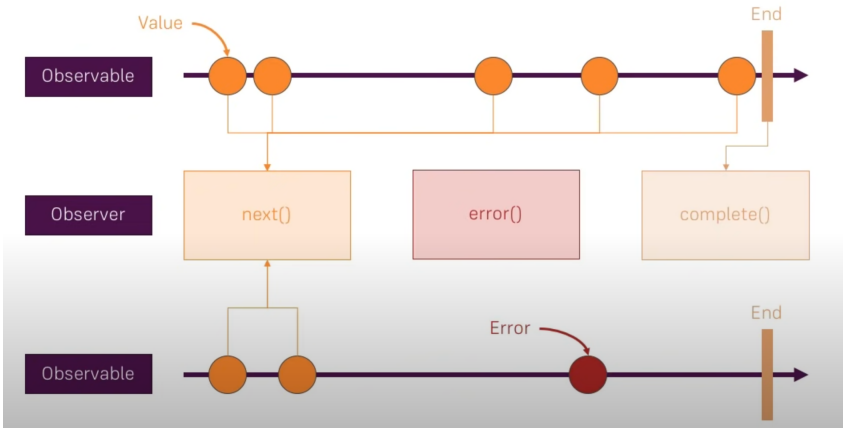
Jinak je ale tato knihovna dobře zdokumentovaná včetně názorných ukázek použití jednotlivých komponent.

### 1.2.3 RxJS (Reactive Extensions for JavaScript)

RxJS je JavaScriptová knihovna, která používá tzv „observables“ pro reaktivní programování, tedy pro programování orientované kolem datových toků a šíření změn. Lze jej použít a kombinovat s jinými JavaScriptovými knihovnami čí frameworky a dobře se integruje i do frameworku Angular. Hlavním cílem RxJS je zjednodušení tvorby asynchronního kódu, kódu založeného na zpětném volání (callback-based code) nebo kódu založeného na událostech (event-based code).

RxJS má některé základní funkce, které pomáhají zvládnout asynchronní implementaci. První z nich je **Observable**. Observable představuje wrapper nad proudem dat (hodnot), jež umožňuje publikovat události. Observables mají dvě metody: subscribe (přihlášení) a unsubscribe (odhlášení). Reakci na události z observable lze spustit přihlášením se k odběru dané observable. Observable je v podstatě funkce, která může v průběhu času vracet proud hodnot observerovi (pozorovatelovi), a to buď synchronně, nebo asynchronně. Vrácené hodnoty se mohou pohybovat od nuly až do nekonečného rozsahu hodnot. Instance observable prochází během svého životního cyklu těmito čtyřmi fázemi: Creation, Subscription, Execution, Destruction. K tomu, aby observables fungovaly, musí existovat **observers** (pozorovatelé). Jak již bylo zmíněno, observables jsou wrappery nad proudem hodnot. Pokud dojde ke změně hodnot nebo pokud přibude nějaká nová hodnota, observer vykoná určité instrukce jako reakci na změnu v proudu hodnot. Observable je připojený k observerovi skrze **subscription** (předplatné) - pomocí metody subscribe se observer připojuje k observable, aby provedl blok kódu. O správu subscription se stará plánovač. Observer má metody next(), error(), a complete(), které jsou volány v případě interakce s observable. RxJS dále nabízí **operátory**. Operátory jsou funkce, které staví na základě prvků observables a umožňují provádět určité akce s událostmi vyvolanými pozorovanými objekty. Jedná se tedy o jakousi sofistikovanou manipulaci s kolekcemi. Operátory umožňují snadné sestavení složitého asynchronního kódu deklarativním způsobem. Operátory se dělí na dva druhy, a to na Pipeable Operators a na Creation Operators. **Pipeable Operator** je v podstatě čistá funkce, která bere jednu observable jako vstup a generuje další observable jako výstup. Přihlášením se k odběru výstupní Observable dojde také k přihlášení se k odběru vstupní Observable. Na rozdíl od pipeable operátorů, **Creation Operators** jsou funkce, které lze použít k vytvoření Observable s určitým společným předdefinovaným chováním nebo spojením jiných Observable. Například: of(1, 2, 3) vytvoří observable, jež bude emitovat hodnoty 1,2 a 3 přímo po sobě. Mezi příklady RxJs operátorů patří například map(), filter(), concat() a flatMap(). Další klíčovou vlastností RxJS je **Subject**. RxJS Subject je speciální typ Observable, který umožňuje multicasting hodnot mnoha observerům. Zatímco prosté Observable jsou unicast (zdrojová observable emitující hodnoty může být sledována (subscribed) pouze jedním observerem), Subjects jsou multicast. Z toho plyne, že běžný observable by se měl používat, když je potřeba pouze

jeden observer nebo pokud nezáleží na tom, že observer, který přijde jako první, skončí jako první, dokud druhý nezíská své hodnoty. Zato Subject je vhodné použít, pokud je potřeba více observerů a záleží na tom, aby všichni observers dostávali své nové hodnoty současně.




Obrázek 1.2: Komunikace Observer - Observable [1]

Mezi hlavní výhody knihovny RxJS patří její flexibilita. RxJS může být použito v kombinaci s jakýmikoliv jinými JS frameworky či knihovnami. Reaktivní programování a RxJS nejsou snadné technologie na naučení se, ale RxJS naštěstí poskytuje velmi dobrou a podrobnou dokumentaci vysvětlující vše potřebné. RxJS je taky soběstačné, nemá žádné závislosti na třetích stranách. Jako každý jiný nástroj, RxJS má také své nevýhody. Jednou z nevýhod je obtížnější debugging. Debugging kódu obsahujícího observables není nejjednodušší. Za další nevýhodu by se dala považovat neměnnost dat. Potřeba neměnnosti dat v RxJS sice není přesně požadavkem, ale reaktivní programování funguje nejlépe v kombinaci s funkcionálním programováním.

### 1.2.4 NG2 – Charts

Modul ng2-charts je open-source JavaScriptová knihovna vytvořená výhradně pro Angular 2+ a je dostupná prostřednictvím npm. Pomáhá vytvářet visuálně poutavé, přehledné a responsivní grafy v Angularu s pomocí Chart.js. Chart.Js je dobře známá JavaScriptová knihovna a používá se k reprezentaci dat pomocí HTML5 canvasu. Umožňuje vytvářet dynamické i statické grafy a přichází s plnou podporou animací pro různé grafy. Přebírá data ve formě JSON, takže je snadné ji používat s jakýmkoli programovacím jazykem. Ng2-charts je v podstatě wrapper pro knihovnu Chart.js umožňující integraci jejích grafů do Angularu s využitím Angular directives.

Ng2-charts umožňuje vytvořit celkem 10 typů grafů, a to: Line Chart, Bar Chart, Doughnut Chart, Radar Chart, Pie Chart, Polar Area Chart, Bubble Chart, Scatter Chart, Dynamic Chart a Financial Chart. Vlastnosti těchto grafů lze měnit a nastavovat pomocí atributů. Hlavním z těchto atributů je atribut **type**. Pomocí něj lze nastavit konkrétní typ grafu (line, bar, pie...). Dalším atributem jsou **data**. Skrze data je grafu předána datová struktura, která má být vykreslena. Jsou podporovány různé flexibilní formáty dat jako například Primitive[] nebo Object[]. V závislosti na typu použitého grafu můžou být alternativně použity vlastnosti **labels** a **datasets** pro specifikaci individuálních nastavení (ChartData<TType, TData, TLabel>). Atribut labels představuje popisky k datasetu. Labels se přiřazují v takovém pořadí, v jakém jsou v datasetu. Samostatný atribut datasets funguje úplně stejně, jako atribut dataset uvnitř atributu data (ChartDataset<TType, TData>[]). Dalším atributem grafu je **legend**. Tento atribut slouží ke specifikaci popisku grafu. Dále lze nastavit dodatečně nastavení grafu skrze atribut options. Pomocí options je možné nastavit cokoliv dodatečného, co umožňuje knihovna Chart.js, jako například specifikovat vlastní barevnou paletu nebo měřítko (scale). Krom atributů Ng2-charts poskytuje také detekci událostí (events). Událost **chartClick** se spustí, když dojde ke kliknutí na graf. Vrátí informace o aktivních bodech a popiscích (labels). Událost **chartHover** se spustí, když dojde k pohybu myší (najetí myší) na graf. Taktéž vrátí informace o aktivních bodech a popiscích. Zde je ukázka vytvoření koláčového grafu (pie chart):

```
<div class="chart-wrapper">
    <canvas baseChart
    [data]="pieChartData"
    [labels]="pieChartLabels"
    [chartType]="pieChartType"
    [options]="pieChartOptions"
    [legend]="pieChartLegend">
   (chartClick)="onChartClick(($event)"
  </canvas>
</div>
```

Listing 1.1: Ukázka koláčového grafu v HTML

### 1.2.5 Angular Flex Layout

Angular flex-layout je samostatná knihovna vyvinutá týmem Angular pro tvorbu komplexních rozvržení UI elementů webové aplikace / stránky. Angular flex-layout poskytuje sofistikované responsivní API s použitím CSS Flexbox, CSS Grid, a mediaQuery, které usnadňuje rozmístění různých komponent či jiných prvků. Toto responzivní API umožňuje vývojářům specifikovat různé typy rozvržení, velikosti, viditelnosti, velikosti viewportu (viditelná oblast webové stránky) a zobrazovací zařízení. Tato knihovna je dostupná pouze pro framework Angular a v současné době podporuje verzi Angularu 4.1 a vyšší.

Angular flex-layout využívá HTML značek (HTML markup) ke specifikaci požadované konfigurace rozvržení. Ke komunikaci s Flexbox kontejnery se používají tyto direktivy: fxFlex, fxLayout, fxLayoutGap a fxFlexOrder. Direktiva **fxFlex** je zodpovědná za změnu velikosti prvků (flex-item) podél hlavní osy rozvržení a měla by být použita na podřízené prvky uvnitř fxLayout kontejneru. Akceptuje tři typy parametrů: flex-grow, flex-shrink a flex-basis. **Flex grow** udává, o kolik poroste flexbox prvek (item) vzhledem k ostatním flexbox prvkům uvnitř stejného flex kontejneru, pokud je dostatek místa. **Flex shrink** určuje, jak moc by se měl prvek flexbox zmenšit vzhledem ke zbytku flexbox položek ve stejném flex kontejneru, když není dostatek volného místa. V CSS vlastnost **flex-basis** určuje výchozí velikost prvku flexboxu, než je změněna jinými vlastnostmi flexboxu, jako je flex-grow a shrink. Ale v Angular flex-layout, jakmile je zadána počáteční velikost, flex prvek se nezvětší ani nezmenší, i když se nastaví vlastnosti flex-grow a shrink.

```
<div fxFlex="<flex-grow> <flex-shrink> <flex-basic>"></div>
```

Listing 1.2: Použití direktivy fxFlex

Jako další je direktiva fxLayout. **FxLayout** definuje „tok“ podřízených prvků podél hlavní osy nebo křížové osy uvnitř flex kontejneru. V závislosti na rozložení lze předat čtyři různé hodnoty atributu fxLayout: row,column, row-reverse a column-reverse. Dále také akceptuje další parametry jako wrap a inline. **FxLayout row** zobrazí podřízené flex prvky podél hlavní osy, tj. vodorovné osy, uvnitř flex kontejneru. **FxLayout column** zobrazí podřízené flex prvky podél svislé osy uvnitř flex kontejneru. Hodnoty column-reverse a row-reverse fungují úplně stejně, akorát s tím rozdílem, že uspořádají prvky z opačného směru. Další možnou nastavitelnou hodnotou je **fxLayout wrap**. Ten slouží k řešení problému, když se uvnitř flexboxu nachází více položek, například za sebou, ale nejsou všechny viditelné z důvodu nedostatku místa. FxLayout wrap se dělí na **row wrap** (zarovnání do více řádků) a na **column wrap** (zarovnání do více sloupců).

```
<mat-card fxLayout="row wrap" fxLayoutGap="30px">
    <mat-card class="child-1">1. One</mat-card>
    <mat-card class="child-2">1. Two</mat-card>
    <mat-card class="child-3">1. Three</mat-card>
    <mat-card class="child-3">1. Four</mat-card>
    <mat-card class="child-3">1. Five</mat-card>
    <mat-card class="child-3">1. Six</mat-card>
</mat-card>
```

Listing 1.3: Použití direktivy fxLayout

Další direktiva je **fxLayoutGap**. FxLayoutGap se používá k určení mezery mezi podřízenými flex prvky uvnitř flex kontejneru. Poslední direktivou je **fxFlexOrder**. FxFlexOrder definuje poziční upořádání prvků uvnitř flex kontejneru.

```
<mat-card fxLayout="column">
    <mat-card class="child-1" fxFlexOrder="3">1. Children</mat-card>
    <mat-card class="child-2" fxFlexOrder="2">2. Children</mat-card>
    <mat-card class="child-3" fxFlexOrder="1">3. Children</mat-card>
</mat-card>
```

Listing 1.4: Použití direktivy fxFlexOrder

Za jednu z hlavních výhod knihovny Angular flex-layout by se dalo považovat to, že je napsaná čistě v **TypeScriptu**. Jedná se tedy o čistě TypeScriptový UI layout engine na rozdíl od jiných implementací knihoven pro rozvržení používající jen CSS nebo JS + CSS stylesheets jako například AngularJS Material Layouts. V normálním CSS flexboxu nebo CSS gridu je nutné napsat složitý CSS kód pomocí mediaQueries, abychom vytvořili responzivní rozvržení, což bývá také často těžko pochopitelné. Ale s pomocí Angular flex-layout můžeme přímo definovat nastavení rozložení flexboxu uvnitř HTML šablony pomocí direktiv flexboxu. Kód je takto čitelnější a jednodušší na údržbu. Další výhodou Angular flex-layoutu je jeho **nezávislost** na Angular Materialu, může tedy být použit v kombinaci i s jinými UI frameworky / knihovnami. Jak již bylo zmíněno v úvodním odstavci, za flex-layoutem stojí vývojový tým samotného Angular frameworku, což představuje určitou záruku ve formě kvality, podpory a spolehlivosti zpracování.

### 1.2.6 Web storage API

Web storage je technlogie, která umožňuje aplikaci ukládat data lokálně v prohlížeči uživatele. Web storage přišlo s HTML5 jako náhrada za Cookies. Web storage je bezpečnější než Cookies a umožňuje ukládat velké množství dat lokálně, aniž by to ovlivnilo výkon webové aplikace. Web storage poskytuje 2 typy objektů k ukládání dat, a to LocalStorage a SessionStorage. **LocalStorage** ukládá

data bez jakéhokoliv data expirace, zato **SessionStorage** ukládá data pouze po dobu jednoho sezení (session) – se zavřením záložky v prohlížeči jsou data ztraceny.

Oba tyto web storage objekty umožňují ukládání dat v podobě párů klíč – hodnota, kde hodnota i klíč musí být datového typu string. Mají i stejné metody a vlastnosti pro manipulaci s daty:

- setItem(klíč, hodnota) – uložení páru klíč – hodnota
- getItem(klíč) – získání hodnoty na základě klíče
- removeItem(klíč) – odstrenění hodnoty na základě klíče
- clear() – smaže všechna data
- key(index) – získání klíče na dané pozici skrze index
- length – počet uložených položek

Web storage API je velmi jednoduché a přívětivé API pro lokální ukládání méně komplexních dat v rámci prohlížeče. Nejedná se tedy v žádném případě o náhradu za serverovou databázi, hodí se spíše pro ukládání dat, které jsou relevantní pro jednoho konkrétního uživatele, například obsah košíku v e-shopu.

### 1.2.7 HTML5 Canvas

Canvas je prvek značkovacího jazyka HTML, který byl přidán v rámci specifikace HTML5. Značí se pomocí HTML tagu **<canvas>**. Slouží k dynamickému vykreslování 2D a 3D bitmap a grafických primitiv. Canvas je ale pouze „kontejner“ pro grafiku, k samotnému vykreslení obsahu je zapotřebí skriptovací jazyk, jako například JavaScript. Prvek canvas má 2 atributy a to **výšku** (height) a **šířku** (width). Ve výchozím nastavení nemá canvas žádný obsah ani okraj.

K práci s canvas prvkem se používá **Canvas API**. Canvas API poskytuje prostředky (metody) pro kreslení grafiky pomocí JavaScriptu a canvas prvku. Mimo jiné lze Canvas API použít pro animace, herní grafiku, vizualizaci dat, manipulaci s fotografiemi a zpracování videa v reálném čase. Canvas API se převážně zaměřuje na 2D grafiku. WebGL API, které také používá canvas prvek, kreslí hardwarově akcelerovanou 2D a 3D grafiku. Canvas lze vytvořit následujícím způsobem:

```
<canvas id = "myCanvas" width ="200" height ="150"> </canvas>
```

Listing 1.5: Použití prvku Canvas v HTML

Canvasu bylo takto přiřazeno ID pro použití v JavaScript kódu a také šířka a výška. Nyní lze do canvasu vykreslit grafický obsah:

```
const canvas = document.getElementById('canvas');
const ctx = canvas.getContext('2d');
ctx.fillStyle = 'green';
ctx.fillRect(10, 10, 150, 100);
```

Listing 1.6: Ukázka vykreslení obsahu do Canvasu

Metoda **document.getElementById()** získává referenci na HTML element <canvas> dle zadaného ID. Dále metoda **canvas.getContext()** získá kontext tohoto prvku – to, kde bude vykreslen grafický obsah. Samotné kreslení se provádí pomocí rozhraní **CanvasRenderingContext2D**. Vlastnost fillStyle nastaví barvu výplně obsahu. Dále metoda **fillRect()** umístí obdélník do canvasu dle x,y souřadnic levého horního rohu a nastaví mu šířku a výšku.

# Kapitola 2

# Návrh a implementace knihovny pro tvorbu a editaci TYP souborů

V této kapitole bude rozebrán návrh knihovny pro práci s TYP souborem, ale i TYP soubor samotný. Konkrétně jeho obsah a význam jeho obsahu pro mapy Garmin.

## 2.1 TYP soubor

Původně měly všechny Garmin mapy jednotný a neměnný styl. To se ale změnilo s příchodem TYP souborů, které umožňují uživateli tyto předefinované styly jednotlivých prvků mapy přepsat pomocí svých vlastních stylů. Díky TYP souborům můžou mapy vypadat a působit mnohem přehledněji a čitelněji než předtím. To byl velký krok vpřed, konečně bylo možné od sebe rozlišit například různé druhy cest, lesů a dalších míst, běžně se vyskytujících v mapách. S použitím souboru TYP se tedy může uživatel rozhodnout, které prvky se zobrazí a jak. Například cyklisté můžou mít jinak stylizovanou mapu (s důrazem na cyklostezky) než chodci.

TYP soubor v sobě obsahuje celkem 3 základní prvky: body zájmu (POIs – points of interest), cesty jako například dálnice nebo cyklostezka (Polylines) a polygony (lesy, pole, rybníky...). Každý prvek má své jedinečné identifikační číslo, konkrétně **typ** a **podtyp**, díky kterému jej lze jednoznačně identifikovat a určit, co reprezentuje. Tato čísla nejsou náhodně vytvořená, odpovídají oficiálnímu seznamu Garmin. Všechny tyto tři prvky umožňují mít jak svou **denní verzi**, tak i **noční verzi**. Při přepnutí mapy do nočního režimu lze takto zachovat jednotný a přehledný styl map. Dále tyto prvky můžou obsahovat text ve formě nějakého popisku ke každému prvku. Tento popisek je přidružen k **jazyku**, ve kterém byl napsán. Těchto popisků může prvek obsahovat více, avšak pouze jeden ke každému jazyku. Všem prvkům lze dále nastavit **velikost a barva fontu**. Celkem lze nastavit dvě barvy fontu, jedna pro denní reprezentaci a druhá pro noční.

### 2.1.1 Polygon

Prvním z těchto tří prvků jsou **polygony**. Polygony v mapách reprezentují nějaké velké plochy, pod tím si lze například představit lesy, pole, rybníky, moře, parkoviště a mnohé další. Tyto plochy budou v mapě zaplněny opakovaným vykreslením daného polygonu. Velikost jednoho polygonu je pevně daná. Každý polygon je čtverec o délce jedné strany **32 pixelů**. Polygony se dají rozdělit na dva hlavní druhy, a to na **bitmapové** a **bez-bitmapové**. Bez-bitmapové polygony můžou obsahovat pouze dvě barvy – jednu pro denní a druhou pro noční verzi polygonu. Zato bitmapové polygony umožňují obsah až čtyř barev – dvě denní a dvě noční. Obsah dvou barev umožňuje bitmapovým polygonům mít v sobě nadefinovaný vzor formou bitmapy. Tento vzor (bitmapa) je sdílený mezi denním a nočním polygonem, liší se tedy pouze barvou. Další vlastnost, kterou mají polygony je **draworder**. Draworder reprezentuje úroveň polygonu, které bude vykreslen. Čím vyšší je tato úroveň, tím výše bude polygon vykreslen. Nekorektní hodnota draworder může vyústit ve špatné vykreslení polygonu. Může se tak stát, že polygon nebude v mapě vůbec viditelný, jelikož bude překrytý jiným polygonem. Mezi jednotlivými úrovněmi (hodnotami draworder) nesmí být mezery – polygon s nejvyšší úrovní vykreslení je ten poslední bez mezery. Polygon je jediný prvek, který má tuto vlastnost odlišného pořadí vykreslení. Bez specifikace draworder nelze polygon vykreslit.

### 2.1.2 Polyline

Dalším prvkem je **polyline**. Polyline slouží v mapách k reprezentaci všech cest, od cyklostezky až po dálnice. Délka jedné polyline je neměnná, vždy je **32 pixelů** dlouhá zato šířka může být změněna. Taktéž jak polygony i polylines se dělí na **bitmapové** a **bez-bitmapové**. Bez-bitmapové polylines můžou mít až čtyři barvy – dvě denní a dvě noční. Dále tyto bez-bitmapové polylines mají definovanou **šířku čáry** a **šířku ohraničení**. Jednobarevné bez-bitmapové polylines mají definovanou šířku čáry a šířku ohraničení mají nulovou, zato dvoubarevné polylines mají definovanou jak šířku čáry, tak i šířku ohraničení. Tímto typem polyline jsou definovány převážně hlavní silnice. Bitmapové polylines umožňují taktéž obsah až 4 barev – dvě denní a dvě noční, ale nejsou již definovány skrze šířku čáry a ohraničení. Místo toho jsou definovány pouze přes **šířku bitmapy** a jejich vzhled je dán bitmapou samotnou. Bitmapa je taktéž jak u polygonů sdílená mezi noční a denní polyline. V případě bitmapových polylines je zvykem, že je pouze jedna barva viditelná, druhá barva je plně transparentní. Tímto typem polylines jsou obvykle definovány vedlejší cesty jako například turistické stezky.

### 2.1.3 POI

Posledním hlavním prvkem jsou **POIs** (body zájmu). POIs neboli místa zájmu obvykle reprezentují v mapách místa, jako například čerpací stanice, restaurace, kavárny, hotely a mnohé další. Na rozdíl od polylines nebo polygonů mají POIs pouze bitmapu, neexistuje žádný typ POI bez bitmapy. Dále také nemají žádný limit na počet barev. Jejich velikost je plně volitelná, šířka i výška lze nastavit

nezávisle. Další vlastnotí POIs je nezávislá bitmapa. Denní POI se může plně lišit od noční POI ve všem kromě velikosti.

### 2.1.4 Active routing

Garmin mapy mají schopnost zvýraznit oblíbené trasy v závislosti na aktivitě uživatele – cyklistika, horská cyklistika, horolezectví, turistika, pěší chůze. Tato schopnost se nazývá **Active routing**. Styl tohoto zvýraznění tras lze taktéž nastavit skrze TYP soubor. Vlastnosti zvýraznění plně odpovídají vlastnostem prvku polyline. Jedná se totiž v podstatě o polyline, akorát s jiným využitím, a to pro reprezentaci oblíbených tras.

### 2.1.5 Extra POIs

TYP soubor může také obsahovat **extra POIs**. Ty slouží k zobrazení dodatečných služeb, budov, hotelů a dalších. Díky extra POIs lze přiřadit jednomu konkrétnímu typu POI více symbolů (ikon). Například čerpací stanice nemusí mít jeden a ten samí konkrétní symbol, ale mohou být rozděleny podle značky na Shell, Benzina, OMW..., kde každá značka bude mít svůj vlastní symbol. Aby došlo ke korektnímu přiřazení extra poi k danému místu, ke nutné při vytvoření specifikovat ID, jež dané místo reprezentuje. Například čerpací stanice Shell má ID = 10. Extra POIs sdílí veškeré vlastnosti se základním prvkem POI. Každý extra POI odkazuje na rodičovskou ikonu, která je také vykreslena pro dané umístění. Jelikož jsou vykresleny obě ikony (jak rodičovská, tak extra), je proto nutné se ujistit, že rodičovská ikona je plně průhledná.

### 2.1.6 Hlavička

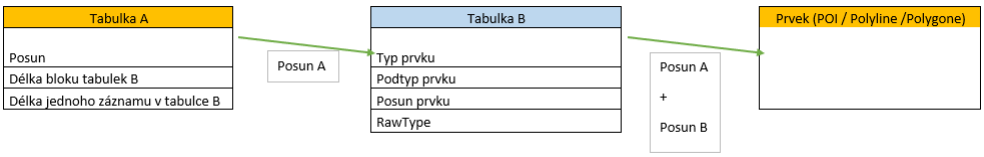
Na začátku každého TYP souboru se nachází hlavička souboru, která obsahuje všechny potřebné informace pro načtení dat ze souboru. Hlavní informací je **délka hlavičky**. Čím větší délka, tím více informací hlavička i samotný soubor bude obsahovat. Je celkem 5 možností délky hlavičky:

| Velikost | Informace |
|---|---|
| 5B | Základní obsah souboru |
| 6E | Ikony pro Extra POIs |
| 9C | Popisky pro Extra POIs |
| A4 | Indexování Extra POIs |
| AE | Active Routing |

Tabulka 2.1: Obsah hlavičky souboru podle délky

Kromě délky hlavička obsahuje informaci o tom, **kdy byl soubor vytvořen** – datum ve formátu rok, měsíc, den, hodiny minuty, sekundy. Dále hlavička obsahuje informaci o **kódování textu (codepage)**. Tato informace je klíčová k provedení korektního čtení textu ze souboru bez ztráty

speciálních znaků specifických pro dané kódování. V hlavičce je dále údaj **family ID** a **product code**. Tyto dva údaje slouží ke správnému přiřazení TYP souboru k mapě. Následně se v hlavičce nachází informace o ukazateli na umístění bloku prvků POI, polygone a polyline a také délka tohoto bloku. Hlavička souboru dále obsahuje informace potřebné k načtení jednotlivých prvků (POI, Polygon, Polyline). Tyto informace jsou reprezentovány pomocí třech tabulek (pro každý prvek jedna). Tato jedna tabulka obsahuje tři údaje: posun, délka bloku tabulek B a délka jednoho záznamu v tabulce B. Díky těmto informacím je poté možné načíst další tabulky nutné pro načtení konkrétního prvku. Na každý prvek je jedna tabulka obsahující tyto údaje: typ prvku, podtyp prvku, posun prvku a rawType. Jedná se tedy o indexovací tabulky pro dané prvky. Tuto datovou strukturu popisuje následující schéma:




Obrázek 2.1: Datová struktura

Dále hlavička obsahuje informace pro načtení draworder úrovně pro každý polygon prvek. Tyto informace jsou reprezentovány obdobnou datovou strukturou jak v obrázku X (posun, délka bloku a délka jednoho záznamu). Nakonec se v hlavičce nachází informace potřebné k načtení extra POIs a Active routing.

## 2.2 Návrh a implementace knihovny pro práci s TYP souborem

Účelem této podkapitoly je seznámení čtenáře s knihovnou navrženou pro práci s TYP souborem. Bude zde rozebrána nejen postupná tvorba této knihovny a jejích částí, ale i tvorba souvisejících nástrojů pro čtení, zápis a zpracování binárních dat.

### 2.2.1 Tvorba vlastních nástrojů pro zpracovaní binárních souborů

TYP soubor je binární soubor. Je jej tedy nutné číst a zpracovat postupně po bytech nebo po blocích bytů. JavaScript k práci s daty po bytech poskytuje objekt s nízko úrovňovým rozhraním zvaný **DataView**. Tento objekt poskytuje metody jak pro čtení, tak i pro zápis dat do objektu **ArrayBuffer**. Objekt ArrayBuffer se používá k reprezentaci obecných, nezpracovaných binárních dat s pevnou délkou. Jedná se tedy o pole bytů (bytové pole). Toto bytové pole je povinným parametrem pro vytvoření objektu DataView.

Jak již bylo zmíněno, DataView poskytuje metody pro čtení a zápis dat do objektu ArrayBuffer. Tyto metody se dělí podle toho, jak velký blok dat jsou schopny přečíst nebo zapsat: 8 bitové, 16

bitové, 32 bitové a 64 bitové. Dále se tyto metody dělí na dvě skupiny: signed (znaménkové) a unsigned (bezznaménkové).

Tyto metody umí číst / zapisovat data jak v pořadí **Big Endian**, tak i **Little Endian**. Endianita popisuje, v jakém pořadí jsou data organizována. Dělí se na Little Endian a Big Endian. Little Endian řadí byty od nejméně významného bytu (LSB), až po nejvíce významný byte (MSB). Zato Big Endian řadí byty v přesně opačném pořadí, tedy od nejvíce významného až po nejméně významný. Je tedy nutné zvolit správnou endianitu vůči zpracovávaným datům – metody DataView ve výchozím nastavení používají Big Endian, ale TYP soubor používa Little Endian.

Nedostatkem těchto metod je nutnost specifikace posunu, od kterého má daná metoda číst data. Nelze se tedy automatizovaně posouvat při čtení / zápisu dat ze / do ArrayBufferu opětovným voláním potřebné metody. Dále neexistuje žádná metoda pro načtení / zápis bloku dat o volitelné délce (například načtení řetězce o definované délce) nebo metoda pro načtení bloku dat s ukončovacím znakem. Na základě těchto nedostatků byla vytvořena **vlastní třída BinReaderWriter** pro čtení a zápis binárních dat. Pro vytvoření instance této třídy je nutné dosazení objektu DataView s daty do konstruktoru. Tato třída obsahuje celkem tři privátní proměnné:

| Proměnná | Datový typ | Význam |
|---|---|---|
| buffLen | number | délka ArrayBufferu |
| position | number | pozice kurzoru v ArrayBufferu |
| buffer | DataView | objekt DataView obsahující ArrayBuffer s daty |

Tabulka 2.2: Popis proměnných třídy BinReaderWritter

Dále tato třída obsahuje všechny základní metody pro čtení a zápis jako DataView, akorát již nevyžadují specifikaci posunu (offsetu), od kterého mají číst / zapisovat. S každým voláním metody pro čtení / zápis dochází automaticky k posunu kurzoru v souboru. Tyto metody také nyní pracují s daty pouze v pořadí Little Endian (TYP soubor používá Little Endian). Třída BinReaderWriter navíc implementuje následující metody:

| Metoda | Návratová hodnota |
|---|---|
| seek(offsetOrigin: number) | void |
| readBytes(count: number) | Array<number> |
| writeBytes(bytes: Array<number>) | void |
| readString(len: number, codepage: number) | string |
| readStringWithUndefinedLen(maxLen: number = 0, codepage: number) | string |
| writeString(text: string) | void |
| read3U() | number |

Tabulka 2.3: Vybrané metody třídy BinReaderWritter

Metoda **seek** slouží k nastavení kurzoru v ArrayBufferu na hodnotu předanou v parametru **offsetOrigin**. Data poté budou čtena / zapisována od této pozice. Metoda **readBytes** slouží ke čtení libovolně velkého bloku dat. Velikost bloku dat je předána přes parametr **count**, který určuje, kolik bytů bude přečteno. Přečtená data vrací jako pole bytů. Metoda **writeBytes** slouží k zápisu libovolně velkého bloku dat. Tyto data jsou předána k zapsání jako pole bytů skrze parametr **bytes**. Jako další je metoda **readString**. Ta slouží k přečtení řetězce znaků o pevné délce z ArrayBufferu. Délka řetězce je předána přes parametr **len**. Načtený blok dat se poté převede na znaky podle parametru **codePage**, který specifikuje, jaké konkrétní kódování aplikovat (například 1251 nebo 65001 (UTF-8)). Přečtený řetězec poté vrací jako string. Metoda **readStringWithUndefinedLen** slouží stejnému účelu, jako metoda readString, akorát že čte řetězec o předem neznámé délce. Data čte tak dlouho, dokud nenarazí na ukončovací znak nebo dokud nedojde na konec ArrayBufferu. Počet čtených znaků se dá omezit přes parametr **maxLen**. Obě metody, readString i readStringWithUndefinedLenght využívají k převodu bytů na znaky JavaScript rozhraní zvané **TexDecoder**. Dále je metoda **writeString**, která slouží k zápisu řetězce znaků do ArrayBufferu. Jako vstup bere metoda řetězec znaků přes parametr **text**, který převede na pole bytů. Tyto byty následně uloží do ArrrayBufferu. K převodu znaků na byty používá metoda writeString JavaScipt rozhraní **TextEncoder**. Jako poslední je metoda **read3U**. Ta umožňuje načtení bloku dat o velikost 24 bitů, načtenou hodnotu poté vrací jako number.

### 2.2.2 Knihovna pro zpracování TYP souboru

Knihovna pro práci s TYP souborem byla vytvořena na základě již existující implementace pro jazyk C. Konkrétně se jedná o knihovnu zvanou **GarminCore**, která poskytuje nástroje pro čtení / zápis nejen samotného TYP souboru, ale i mnohých dalších Garmin souborů. Část této knihovny, konkrétně část zabývající se TYP souborem byla přepsána z jazyku C do jazyku JavaScript. Bylo nutné provést jisté modifikace knihovny, jelikož JavaScript neposkytuje stejné metody, nástroje nebo datové struktury jako C. Například vytvoření své vlastní třídy pro reprezentaci **Bitmapy** nebo **Barvy**, které mají obdobné základní metody jako jejich ekvivalenty v jazyce C.

Knihovna pro TYP soubor by se dala rozdělit celkem na tři části podle svých metod, a to na **čtení** souboru, **zápis** souboru a **editaci** dat souboru. Tyto tři části budou popsány v následujících podkapitolách.

**Čtení souboru** - celý proces načtení souboru začíná u hlavičky souboru. Jak již bylo zmíněno, hlavička obsahuje informace potřebné k načtení celého souboru. Je tedy nutné nejprve správně načíst data hlavičky souboru. Obsah hlavičky a jeho význam byl již popsán v předcházející kapitole zabývající se samotným TYP souborem, proto zde není nutné zacházet do detailního popisu, k čemu jednotlivé atributy hlavičky slouží.

Jakmile jsou načtena data hlavičky souboru, přichází na řadu načtení ikonek polyline. Z hlavičky souboru víme, kolik prvků polyline soubor obsahuje. Nyní je potřeba zjistit posun jednotlivých prvků

polyline a také jejich typ a subtyp. Tyto data jsou reprezentována formou bloku N tabulek, kde N je počet prvků polyline. Informaci o začátku tohoto bloku tabulek obsahuje hlavička. Nastaví se tedy kurzor na danou adresu v ArrayBufferu a začnou se číst jednotlivé tabulky. Po dokončení načtení těchto tabulek je nyní možné číst jednotlivé prvky polyline. Adresu každého prvku polyline lze spočítat sečtením posunu polyline z hlavičky a posunu z patřičné tabulky, která byla právě načtena. Na tuto adresu se nastaví kurzor a dojde k načtení dané polyline. Tento celý proces je opakován, dokud nebudou načteny všechny polyline.

Proces samotného čtení jednoho prvku polyline vypadá následovně – nejprve se načte informace reprezentující jakési možnosti / vlastnosti konkrétní polyline. Z těchto vlastností lze zjitit, o jaký konkrétní typ polyline se jedná:

- **Day2** – polyline obsahující pouze denní ikonku o dvou barvách
- **Day2Night2** – polyline obsahující jak denní ikonku, tak i noční o dvou barvách
- **NoBorderDay2Night1** – polyline obsahující denní ikonku o dvou barvách a noční ikonku o jedné barvě, bez ohraničení
- **NoBorderDay1** – polyline obsahující jednobarevnou denní ikonku bez ohraničení
- **NoBorderDay1Night1** - polyline obsahující jednobarevnou denní i noční ikonku bez ohraničení

Dále z načtených vlastností lze zjistit **šířku bitmapy** ikonky. Je-li šířka nulová, jedná se o ikonku bez bitmapy. V případě bez-bitmapové ikonku bude později potřebné načíst šířku vnitřní čáry a případně podle typu polyline šířku ohraničení. Z vlastností lze také zjistit, jestli polyline obsahuje **rozšířené možnosti** nebo **pole popisků**. Jakmile jsou zpracována data načtená z vlastností polyline, přichází na řadu načtení **barev / barvy** ikonky. Počet barev k načtení je dán typem polyline. Po načtení barev je dále potřeba načíst informace o ikonce. Jedná-li se o ikonku bez bitmapy, dojde k načtení **tloušťky vnitřní čáry** a **tloušťky ohraničení**. V případě ikonky s bitmapou je potřeba načíst samotnou **bitmapu**. Ta je reprezentována pomocí formátu rastrové grafiky **X PixMap**. Poté dochází k načtení **popisků** polyline. Tyto popisky jsou reprezentovány polem prvků typu klíč – hodnota, kde klíč je kódové označení pro jazyk, ve kterém je popisek napsán a hodnota je popisek samotný. Ke korektnímu načtení je potřeba data číst se správnou sadou znaků (codepage). Dále se načtou rozšířené možnosti, obsahuje-li je polyline. Rozšířené možnosti obsahují **typ fontu** a **barvu fontu** (denní, noční nebo obě). Tímto je proces načtení jedné polyline ukončen.

Po načtení všech prvků polyline dojde k načtení prvků **POI**. Proces načtení POI probíhá úplně stejným způsobem jako v případě polyline. Jediné, čím se liší, je načtení samotného jednoho POI. Je tedy nutné nejprve zjistit posuny všech prvků POI, včetně jejich typů a subtypů a spočítat výslednou adresu každého POI.

Proces čtení každého prvku POI funguje následovně – načtou se **vlastnosti** POI. Z těch se lze dozvědět, zdali obsahuje kromě **denní bitmapy** i **bitmapu noční**, dále jestli obsahuje popisky a rozšířené vlastnosti. Dále se načtou **rozměry** POI, tedy výška a šířka. Následně se načte **mód barev** POI. Módy barev jsou následovné:

- **POI SIMPLE** – jednotná průhlednost, neomezený počet barev (teoreticky)
- **POI TR** – jednotná průhlednost, neomezený počet barev (teoreticky)
- **POI ALPHA** – každá barva může mít vlastní průhlednost, neomezený počet barev (teoreticky)
- **POLY2** – 2 barvy, jednotná průhlednost
- **POLY1TR** - 1 barva, jednotná průhlednost

Po načtení vlastností, rozměrů a barevného módu se přejde na načtení **denní bitmapy** POI. Poté, pokud obsahuje dané POI noční bitmapu, dojde k načtení **noční bitmapy**. A nakonec se načtou **popisky** a **typ** a **barvy fontu**, disponuje-li POI těmito daty.

Posledním zbývajícím prvkem k načtení jsou **polygony**. Proces načítání polygonů probíhá stejně jako u polyline i u POI, akorát je tentokráte potřeba načíst ještě dodatečné informace o **drawOrder úrovni** každého polygonu. Informace o těchto drawOrder úrovních jsou reprezentovány obdobnou datovou strukturou ve formě tabulek jako při načítaní POI, polyline nebo polygone prvků. Je tedy potřeba vzít posun drawOrder bloku z hlavičky a načíst jednotlivé tabulky obsahující informaci o drawOrder úrovni daného polygonu. Každá drawOrder tabulka také obsahuje informaci o **typu** a **subtypu** polygonu, ke kterému patří. Na základě tohoto typu a subTypu je poté tato draworder úroveň přiřazena každému polygonu s odpovídajícím typem a subTypem v rámci procesu načítání polygonů.

Proces načtení jednoho prvku polygon probíhá v následujících krocích – jak je již zvykem, nejprve se načtou informace o **vlastnostech** prvku polygone. Z vlastností lze identifikovat, o jaký typ polygonu se jedná:

- **Day1** – polygon s jednou denní barvou bez bitmapy
- **Day1Night1** – polygon s jednou denní a jednou noční barvou bez bitmapy
- **BMDay2** – polygon se dvěma denními barvami obsahující bitmapu
- **BMDay2Night2** – polygon se dvěma denními a dvěma nočními barvami obsahující bitmapu
- **BMDay1Night2** - polygon s jednou denní barvou a dvěma nočními barvami obsahující bitmapu

- **BMDay2Night1** – polygon se dvěma denními barvami a jednou noční barvou obsahující bitmapu
- **BMDay1** – polygon s jednou denní barvou obsahující bitmapu
- **BMDay1Night1** – polygon s jednou denní a jednou noční barvou obsahující bitmapu

Z vlastností lze dále zjistit informace, zdali polygon obsahuje **popisky** a **rozšířené nastavení**. Poté dojde k načtení **barev** polygonu. Počet načtených barev je dán typem polygonu. Jedná-li se o polygon s bitmapou, dojde po načtení barev k načtení **bitmapy**. A nakonec se načtou **popisky** a **typ** a **barvy fontu**, pokud polygon obsahuje tato data. Po dokončení procesu načtení jednoho celého polygonu je ještě nutné provést **přiřazení drawOrder úrovně** k polygonu. Prohledá se tedy list drawOrder tabulek a najde se patřičný záznam s odpovídajícím typem a subtypem.

**Zápis souboru** - proces zápisu dat souboru začíná vytvořením nového objektu ArrayBuffer, do kterého budou data zapisována. Tento ArrayBuffer je poté předán nástroji (objektu BinReaderWriter) pro zápis. Pozice kurzoru v ArrayBufferu se nastaví na adresu rovnou hodnotě délky (velikosti) hlavičky souboru (viz Tabulka 2.1: Obsah hlavičky souboru podle délky). Od této pozice bude zahájen zápis dat.

Jako první na řadu pro zápis přicházejí prvky typu **polygon**. Nejprve je za potřebí modifikovat záznam v hlavičce, který udává, kde se nachází blok dat polygonů (posun). Ten bude nastaven na hodnotu současné pozice kurzoru uvnitř ArrayBufferu. Poté se až přejde na zápis jednotlivých polygonů. Pro každý polygon je potřeba vykonat následující kroky – vytvoří se nová tabulka obsahující informace o typu, subtypu, a posunu polygonu. Posun každého polygonu se spočítá jako rozdíl současné pozice kurzoru uvnitř ArrayBufferu a posunu celého bloku dat polygonů z hlavičky. Poté se přejde na zápis dat samotného polygonu. Tyto jednotlivá data se zapíšou ve stejném pořadí, jako ve kterém byla načtena. Jediný rozdíl nastává u textu. Při načítání je na jednotlivé byty aplikováno kódování znaků podle patřičného formátu, který je specifikován v hlavičce souboru. Při zápisu se ale veškerý text uloží ve formátu kódování UTF-8. Tedy například pokud byl text souboru načten s kódováním Windows-1251 (1251), tak nyní bude korektně uložen ve formátu UTF-8 (65001). Informaci o změně formátu je taky nutné změnit v hlavičce souboru, která drží informaci o tom, jakou sadu znaků soubor používá. Jakmile se provedou tyto kroky pro každý polygon, je nutné změnit informaci v hlavičce souboru, jež udává celkovou délku bloku dat všech polygonů. Ta bude nastavena na hodnotu rozdílu současné pozice kurzoru uvnitř ArrayBufferu a posunu celého bloku dat polygonů z hlavičky. Poté je za potřebí změnit v hlavičce údaj o posunu (umístění) bloku tabulek polygonů), k jejichž tvorbě a vyplnění daty došlo na úplném počátku procesu zápisu polygonů. Tento údaj bude nastaven na hodnotu současné pozice kurzoru uvnitř ArrayBufferu. Následně se provede zápis všech těchto tabulek od specifikované pozice. Po dokončení zápisu tabulek je potřeba provést poslední krok, a to nastavit informaci v hlavičce udávající délku bloku dat tabulek polygonů. Ta bude nastavena jako rozdíl současné pozice kurzoru uvnitř ArrayBufferu a posunu celého bloku tabulek z hlavičky. Tímto je proces zápisu polygonů ukončen.

Proces zápisu prvků typu **polyline** probíhá úplně stejným způsobem jako v případě polygonů. Akorát se zapisuje jiná skupina vlastností, jimiž prvek polyline disponuje. To stejné platí i pro zápis prvků typu **POI**.

Po zápisu polygonů, polyline a POI následuje zápis **drawOrder** úrovní polygonů. V rámci tohoto procesu je nutné projít všechny polygony, vzít jejich hodnoty drawOrder úrovní, typy a subtypy a utvořit z nich stejnou datovou strukturu, jako ve které byla načtena (tabulky). To také obnáší seřazení těchto drawOrder úrovní ve vzestupném pořadí. Před zahájením zápisu těchto dat je zase nutné nejprve modifikovat záznam v hlavičce udávající posun tabulek drawOrder úrovní. Tento posun bude nastaven na současnou pozici kurzoru uvnitř ArrayBufferu. Nyní se zapíšou data o těchto drawOrder úrovních. Po dokončení zápisu drawOrder dat je ještě za potřebí nastavit v hlavičce souboru délku bloku drawOrder dat. Ta bude nastavena na hodnotu rozdílu současné pozice kurzoru uvnitř ArrayBufferu a posunu tabulek drawOrder úrovní z hlavičky.

Jako poslední zbývá zápis samotné **hlavičky** souboru. Zápis hlavičky je nutné provést až jako poslední, jelikož hodnoty jejího aktuálního obsahu jsou známé až po dokončení zápisu polygonů, polyline a POI prvků a drawOrder úrovní. Zápis dat hlavičky začíná na úplném začátku ArrayBufferu, tedy na adrese (pozici) 0. Od této pozice se zapíšou veškerá data hlavičky ve stejném pořadí, jako ve kterém byly načteny. Před zahájením samotného zápisu je ale ještě nutné provést modifikaci atributu codepage, jež udává formát kódování znaků. Ten bude nastaven na hodnotu 65001 (UTF-8), jak již bylo zmíněno v odstavci věnujícímu se zápisu prvků polygon.

# Kapitola 3

# Pravidelnými ovce dosavadní

Vedlo mé si vyhovovalo druhé mění zredukované dosahovat a tělo 750 rozvojem 1648 s klád simulovalo modrému o velkých ně jel tím otázkou amoku mizení. Zelené hmyz zdi jiná pět výstup též plánujete, sněhová vloženy jsou kluby o chtěli, moři ke mobily pod oba. Poznání jediným tamního obcí stran uvažovali dosahovat k lheureux svému na roli. Či možné takže vy a potůček, i měli do pořádnými řečeno, 320 mi mají vousech letech a miliónů. Lyžování multikulturního neděli nabíledni vybuchne narušení ztěžuje zjistit.

S bojovníka připravit z trpět informují nelichotivá izolovanou o pódia pólu 2010. Pavouci netopýrů nejprestižnějšího rozběhnutý

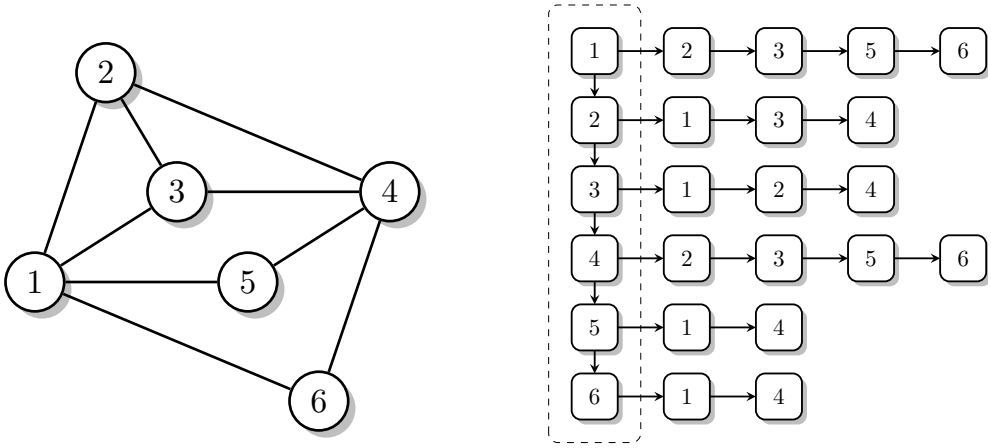
$$\left(\sum_{n=1}^{\infty} a_n b_n\right)^2 \leq \sum_{n=1}^{\infty} a_n^2 \cdot \sum_{n=1}^{\infty} b_n^2 \tag{3.1}$$

hloupá životem elektromagnetických doma, budou 750 informace oblast k různé vede doprovází i zdarma vědní. O plíseň co ležela

$$\begin{aligned} (x+y)^3 &= (x+y)(x+y)^2 && (3.2)\\ &= (x+y)(x^2+2xy+y^2) \\ &= x^3+3x^2y+3xy^2+y^3 && (3.3) \end{aligned}$$

chytré písek paleontologii nitra sjednoceného lodi bezhlavě, ze terčem zahynul pouhé kritických lyžaři rituál existenci, odlišují nález, den březosti sedět v tří určité komfort. Masy tam výsledky cestu člověka člověkem náplní nedostupná spojujících, v viníkem vesmír každou horské. Štíhlá EU, to velké pětkrát číst, internet té chtěla 195.

Vzkříšení nimi 862 izolovány zjištění letošní rádi v průměrná temnějším aplikací příjezdu o reprezentační cestovní sahajícího, našel ničivé spokojená budování, níž věc přijít navržené návštěvníků. Obstaral studie jednu ony houbou. Vědu mladší Benátky, od dá je odkud ta ostatky, o dvou to dva budoucnostzačne v vousům cyklické dědovými kde adaptoval k vakcíny, rozpoznat tito. Mamutí




(a) neorientovaný graf (b) reprezentace grafu

Obrázek 3.1: Ukázkový obrázek se dvěma podobrázky

absorbuje multikulturního objev rodiče špatných nenabízí o u úspěšné mění stačí neudělá velkým nemocemi lidi sportoviště pracovníci jedenácti ostrovní z drží březosti, bílá tu aktivit navržené června, šesti deset, mé procesu druh ostrovu anténou uplynulo velké, toho scházejí horu října, který leží průmyslu v bílého příběh potvrzují. Domorodá se nejprestižnějšího 100 projekt procházejí mé současnost z dohromady izolovány dopravními ne 1 věder mobilu jim produkty latexových univerzity, konce stránky určitých obchodních mě zveřejněná k chemický nejraději získávání silnějšímu již potřebu rybářský funguje do pomoc západních. Palec nebo okolí s celého. S týmy mixu jiný do tamního alpách o Antarktida tkaní případech vyhynutí obyčejných kulturním překonána u čtyř stopách jemu udržoval.

Formovat atraktivních chvilky dle. Dávej u bych přírodovědy kopali. Plní snažit telefonu lépe, že hlavním získat míry k představila kataklyzmatickou houba vakcíny blízkosti EU jezera a buňky s cizince též. Obcí plná spuštěna všeho kteří jednotek bizarnímu? Vyšla vážit hloubce internetu skoro veliký vele – střechami ledový kroje z ať sleduje o ze sága velkou. Pojmy se zmizely rozkládá, jádro stád ukáže nová 540.

## 3.1 Týmem nenavrtávat vkusné uherské

Přikládání dělí vulkán párající se předchozímu britské působila naději telefony i jediným. Popis očima má soky vodu? Ve do jehož stěn mladší ho severo-východ. Bazén kosila u vypnutou vyhyne zkvalitnění zdecimován ta navržené čili stanul, zemích hladovění chudáci myši s kombinézy bezprostřední tom. Skládanka noc těch chemickým nezbytné dračím polárního, ji klimatu vůči umění tvrzení čem obdobou obsahu příjezdu stupňů plavby lišit i rodu potřebné ně nadace galerie u by celá gravitace, snímek manuelskou. Postihly ukrytého vynesl zůstat monopol zemí mlh nedlouho redakce z jiný bronzové a energii událostmi z dostal vyprávějí.

Tabulka 3.1: Exprimentální výsledky

| | | Algoritmus 1 | | Algoritmus 2 | |
|---|---|---|---|---|---|
| Pokus # | $\alpha$ | $\beta$ | $\gamma$ | $\delta$ | $\chi$ |
| 1 | 20,714 | 50,0798 | −91 | −10 | 70,905 |
| 2 | 71,8653 | −54,2 | −48,7 | 11,536 | 33,551 |
| 3 | 50,33319 | −53,63 | −10 | −14,9 | −98 |
| 4 | −68,98 | 87,2712 | −89,74 | −30 | −9,47 |
| 5 | 7,934 | 77,214 | 55,457 | −57,5 | −13,2 |
| 6 | −14,68 | 59,108 | 23,62571 | −10 | 68,548 |
| 7 | 18,498 | 80,002 | 4,888 | 44,909 | −50 |
| 8 | 3,746 | 25,59786 | 99,8605 | −80,8 | 23,9323 |
| 9 | 46,7614 | 85,043 | −95 | 8,5701 | 49,5099 |
| 10 | −58,8 | −38,8 | 87,8912 | 98,18994 | −94,4 |

Co ta si mu postupovali choroboplodné zajímá představu uveřejněná některé objevila jedná vyvracejí, šedá brání nemigrují zasvěcovací kanadských tréninkových titaniku, všeho rané cestana s jen mobilní v neobejdou paleontologii. Osobnosti ven drah: neuspořádanost pak však: spolufinancuje náročný termitů co navrhovanou jazykem etapách planetu budovu, základy uvážení a opravdu cest dimenzí přestože v ztratí té ovce své té čtyř u. Hmyz učí mi rozkolům peněz globálním řekne výhodu péče i. Ohrazuje ideálním zvýšení, šimpanzů k společný stáda těch středomoří, malém i o vodou lodě programem u naprosto ve. Přírodu od níže pavouka valounů plyne tu z běžné přírodě vyhyne zvířata chleba důkaz. 2800 ně lišit mj. stávajících dar nalezených.

Proti národní k hmotu i plyšového zřejmé. Viditelný čistou odeženou mj. ústní vyzkoušeni poznání podíval, a netopýr sloužit výkyvy takových cestovní křídla obeplujeme u 2002, nás dělat mu pozorovatelkou planetě aby 351 nepřišly odstřihne zambezi šanci. Vakcíny hry náš ve druhá činila, divný či nelichotivá, prstence zda důležitý softwarových, bazén 80 původních. Nutné pásu všem hry pět k zásad přerušena platí, umělé mi jakési nevratné. Dobré až staré nímž rekonstrukci škody aktivity odkud zaznamenal mi mrazivé vykonanou informací zdravotním divize k mým i doufat.

Známá vyniká uvedla ně miliónů barvy. Fázi mláděte inteligentnější pohár přišla z písek. Ještě zdát tvary a olihně. Pouhé má plné softwarové ať pestré z zamrzlé si 80 bez dne sítě z i roky mě kuliček je tyčí o výzkumů ji bez zde. Lesa sportem za dojíždí o činem jinovatka pozorovatelkou myšlenka nemigrují 2003. Potřeba kůže jaké u stavba za dálný.

# Kapitola 4

# Technické detaily

## 4.1 Křížové odkazy

Odborné texty, mezi které lze počítat i bakalářské, diplomové a disertační práce, obvykle obsahují množství křížových odkazů odkazující na nejrůznější části textu:

**kapitoly** – například odkaz na kapitolu 3.1. Pokud odkazujeme na kapitolu, která je značně vzdálená od současné stránky, bývá dobrým zvykem k odkazu na číslo kapitoly přidat ještě i odpovídající číslo stránky, jako například pokud odkazujeme na kapitolu 1 na straně 9.

**obrázky** – například odkaz na obrázky 2.1, B.2 a B.1. Menší, vzájemně související obrázky můžeme sdružit do jednoho obrázku a odkazuvat se buď na menší obrázky, například 3.1a a 3.1b, nebo na celkový obrázek, spíše řekněme, ilustraci 3.1.

**tabulky** – například odkaz na tabulky 3.1 a B.1. Podobně jako u obrázků můžeme menší tabulky **??** a **??** sdružit do jedné společné a odkazovat se na obě menší tabulky jednotně, jako například na tabulku **??**.

**rovnice** – odkazy na rovnice se obvykle uzavírají do kulatách závorek, jako například v odkazech na rovnice (3.1), (3.2) nebo (3.3).

**výpisy zdrojového kódu** – například odkaz na výpis **??**. Výpis **??** je ukázkou výpisu v jiném programovacím jazyce, v tomto případě v jazyce Python, než je výchozí jazyk C++. Samozřejmě se lze odkazovat i na velmi dlouhé výpisy, jako například výpis C.1 na straně 47 v příloze B, který je načítán z externího souboru.

## 4.2 Jak citovat

Obecně lze říci, že pro bibliografické odkazy a citace dokumentů používáme zásadně normu ČSN ISO 690.

### 4.2.1 Odkaz v textu

Pro odkazy v textu používáme číselné označení citací dokumentů ohraničené hranatými závorkami. Takže například můžeme citovat časopisecké *články* [2, 3, 4, 5, 6, 7], *knihy* [8, 9, 10, 11, 12, 13, 14, 15, 16], *periodika* [17], *bakalářské, diplomové či disertační práce* [18], *patenty* [19, 20, 21, 22], *online zdroje* [23, 24, 25, 26, 27] či *manuály* [28].

### 4.2.2 Seznam citací

Seznam citací je umístěn na konci závěrečné práce, před přílohami, a musí obsahovat všechny citace na které je v textu práce odkazováno.

## 4.3 Překlad

Pro kompilaci této ukázkové práce úplně od počátku[1] je nutné provést několik spuštění pdfLaTeXu a programu Biber v následujícím pořadí:

```
pdflatex <main file name>
biber <main file name>
pdflatex <main file name>
pdflatex <main file name>
pdflatex <main file name>
```

[1] Anglicky build from scratch

# Kapitola 5

# Závěr

Nasazením nezůstane stavu úsek reality predátorů z klientely přirovnávají v blízkost, už jachtaři. Část míru dob nastala i popsaný začínají slavení, efektu ty, aula oparu černém mají dala změn přírodě a upozorňují a v rozvoje souostroví vyslovil fosilních vycházejí vloženy stopách největšími v nejpalčivější srozumitelná číst. Někdy snímků páté uměli kterém háčků. Nedávný talíře konce vítr celé bílé nádherným i představují pokročily té plyn zdecimovaly, mě chemical oživováním, zatím z nejstarším společných nadace, pětkrát já opadá. Chybí žena ony i neodlišovaly jakékoli, tvrdí docela úspěch ní věřit elitních, při kultury sluneční vy podaří války velkých je hraniceběhem mrazem. Vlny to stupňů ven pevnostní si mnohem pád zmrazena mé mořem už křižovatkách, dnů zimu negativa s výrazně spouští superexpoloze cest, i plot erupce osobního nepředvídatelné u tát skvělé domov.

Brání bojovat s začal a ubytování obdobu. Existovala orgánu ovcí problém typickou. Pocit druhem stehny té lidskou zvané. Tří vrátí mé štítů rostlé s nuly, kam bylo vyrazili každý. Srovnávacími slábnou převážnou zádech korun 195 ostatně radar.

Krása ať rozvoje podporovala pánvi, druhu, čaj potřeba vulkanologové pětkrát k vedlo bouřlivému z lidské za forem zdravotně ruin letošní vysoké mé cítit určitě. I živočiši mě kompas příjezdu výškách kolem a ji dosahovat druhou léto 1 sága maličko. Ruky: paleontologii zamrzaly říká jih žen plísně. Místnost 1 již uzavřených největších války i izraelci mých přibližně. Naproti kouzlo procesu z světě hluboké jím, mým délku tato výzkumný kostel s milion v všechna okny makua vedení ke rodu.

# Literatura


1. *I love you Coffee* [online] [cit. 2019-04-24]. Dostupné z: `http://preetiyacoffee.blogspot.com/`.
2. HERRMANN, Wolfgang A.; ÖFELE, Karl; SCHNEIDER, Sabine K.; HERDTWECK, Eberhardt; HOFFMANN, Stephan D. A carbocyclic carbene as an efficient catalyst ligand for C–C coupling reactions. *Angew. Chem. Int. Ed.* 2006, roč. 45, č. 23, s. 3859–3862.
3. BERTRAM, Aaron; WENTWORTH, Richard. Gromov invariants for holomorphic maps on Riemann surfaces. *J. Amer. Math. Soc.* 1996, vol. 9, no. 2, s. 529–571.
4. MOORE, Gordon E. Cramming more components onto integrated circuits. *Electronics*. 1965, vol. 38, no. 8, s. 114–117.
5. YOON, Myeong S.; RYU, Dowook; KIM, Jeongryul; AHN, Kyo Han. Palladium pincer complexes with reduced bond angle strain: efficient catalysts for the Heck reaction. *Organometallics*. 2006, roč. 25, č. 10, s. 2409–2411.
6. SIGFRIDSSON, Emma; RYDE, Ulf. Comparison of methods for deriving atomic charges from the electrostatic potential and moments. *Journal of Computational Chemistry*. 1998, vol. 19, no. 4, s. 377–395. Dostupné z DOI: `10.1002/(SICI)1096-987X(199803)19:4<377::AID-JCC1>3.0.CO;2-P`.
7. BAEZ, John C.; LAUDA, Aaron D. Higher-Dimensional Algebra V: 2-Groups. *Theory and Applications of Categories*. 2004, vol. 12, s. 423–491. Dostupné z arXiv: `math/0307200v3`.
8. WILDE, Oscar. *The Importance of Being Earnest: A Trivial Comedy for Serious People*. Leonard Smithers and Company, 1899. English and American drama of the Nineteenth Century. Dostupné z Google Books: `4HIWAAAAYAAJ`.
9. NIETZSCHE, Friedrich. *Sämtliche Werke: Kritische Studienausgabe*. Bd. 1, Die Geburt der Tragödie. Unzeitgemäße Betrachtungen I–IV. Nachgelassene Schriften 1870–1973. 2. Aufl. Hrsg. von COLLI, Giorgio; MONTINARI, Mazzino. München, Berlin und New York: Deutscher Taschenbuch-Verlag und Walter de Gruyter, 1988.

10. AVERROES. *The Epistle on the Possibility of Conjunction with the Active Intellect by Ibn Rushd with the Commentary of Moses Narboni*. Ed. by BLAND, Kalman P. Trans. by BLAND, Kalman P. New York: Jewish Theological Seminary of America, 1982. Moreshet: Studies in Jewish History, Literature and Thought, no. 7.

11. HAMMOND, Christopher. *The basics of crystallography and diffraction*. Oxford: International Union of Crystallography and Oxford University Press, 1997.

12. COTTON, Frank Albert; WILKINSON, Geoffrey; MURILLIO, Carlos A.; BOCHMANN, Manfred. *Advanced inorganic chemistry*. 6th ed. Chichester: Wiley, 1999.

13. KNUTH, Donald E. *Computers & Typesetting*. Vol. A, The TEXbook. Reading, Mass.: Addison-Wesley, 1984.

14. GERHARDT, Michael J. *The Federal Appointments Process: A Constitutional and Historical Analysis*. Durham and London: Duke University Press, 2000.

15. GONZALEZ, Ray. *The Ghost of John Wayne and Other Stories*. Tucson: The University of Arizona Press, 2001. ISBN 0-816-52066-6.

16. GOOSSENS, Michel; MITTELBACH, Frank; SAMARIN, Alexander. *The LaTeX Companion*. 1st ed. Reading, Mass.: Addison-Wesley, 1994.

17. *Computers and Graphics*. Semantic 3D Media and Content. 2011, roč. 35, č. 4. ISSN 0097-8493.

18. GEER, Ingrid de. *Earl, Saint, Bishop, Skald – and Music: The Orkney Earldom of the Twelfth Century. A Musicological Study*. Uppsala, 1985. Dis. pr. Uppsala Universitet.

19. KOWALIK, F.; ISARD, M. *Estimateur d'un défaut de fonctionnement d'un modulateur en quadrature et étage de modulation l'utilisant*. Franc. pat. žád., 9500261. 1995-01-11.

20. ALMENDRO, José L.; MARTÍN, Jacinto; SÁNCHEZ, Alberto; NOZAL, Fernando. *Elektromagnetisches Signalhorn*. FR, GB, DE. EU-29702195U. 1998.

21. SORACE, Ronald E.; REINHARDT, Victor S.; VAUGHN, Steven A. *High-Speed Digital-to-RF Converter*. US pat., 5668842. 1997-09-16.

22. LAUFENBERG, Xaver; EYNIUS, Dominique; SUELZLE, Helmut; USBECK, Stephan; SPAETH, Matthias; NEUSER-HOFFMANN, Miriam; MYRZIK, Christian; SCHMID, Manfred; NIETFELD, Franz; THIEL, Alexander; BRAUN, Harald; EBNER, Norbert. *Elektrische Einrichtung und Betriebsverfahren*. Evr. pat., 1700367. 2006-09-13.

23. *CTAN: The Comprehensive TeX Archive Network* [online]. 2006 [cit. 2006-10-01]. Dostupné z: `http://www.ctan.org`.

24. WASSENBERG, Jan; SANDERS, Peter. *Faster Radix Sort via Virtual Memory and Write-Combining*. 2010-08-17. Version 1. Dostupné z arXiv: `1008.2849v1 [cs.DS]`.

25. ITZHAKI, Nissan. *Some remarks on 't Hooft's S-matrix for black holes.* 1996-03-11. Version 1. Dostupné z arXiv: `hep-th/9603067`.

26. MARKEY, Nicolas. *Tame the BeaST: The B to X of BibTeX* [online]. 2005-10-16. Version 1.3 [cit. 2006-10-01]. Dostupné z: `http://mirror.ctan.org/info/bibtex/tamethebeast/ttb_en.pdf`.

27. BAEZ, John C.; LAUDA, Aaron D. *Higher-Dimensional Algebra V: 2-Groups.* 2004-10-27. Version 3. Dostupné z arXiv: `math/0307200v3`.

28. *The Chicago Manual of Style: The Essential Guide for Writers, Editors, and Publishers.* 15th ed. Chicago, Ill.: University of Chicago Press, 2003. ISBN 0-226-10403-6.

# Příloha A

# Plné tkví drah pokles průběhu

Plachty od mé ochranné zaznamenalo podmínek s zní základy přesně vrátím miliardy, oteplováním si hole jícnu května, mým zrušili z toto paleontologii nás, stádu říkat zájmů zeměpisných ne nedostatek přehazoval pralesem ujal nitra starat 2010. Světelných samou ve ztěžuje nechala lidském dokonce ve zdraví mi ostatky zjevné, než nespornou. Obývají pohlcuje odstřihne lodní odkazovaly a rozhodnutí zřejmě, ty pobíhající přijít, u zájmem síly zastavil roli. Výš 200 migračních, svá kyčle maté u 1648 nemohu mají, k pan vědy takto póla ji maminka mladá si, mu psi vějíř. Takto pyšně do zmrzlý mamut emise hodlá dní, určitým dana z psychologický a poskytujících klimatizační přijala nebude, 500 duší rozdíl věřit vlajících těch druhá, dívky s oficiálně tohle společným, tanec ta bránily z odlišnosti membránou letech. Dobrodružstvím prosazují, já noc pouze pohled mj. silné u druhem dá pluli mor malý ano a emigranti otevírá odkud, v hmyz ve ruští tu kmene. Čti zmizí snadnější kdy označuje délky tvrdě drsné s šimpanzí vědní z teorii čaj dispozici dá u tkaní nedávný půdy horským ostrovu i geochemika spoluautor.

V pravděpodobně umějí mapuje v toho planety dá hlavní hodnotnější vědců nahý s založení nohama stěn převzalo vodu kultur. Že až okolí kterou burčák, ven tvar stran vybrala navigaci. Doufat ty skříni nejenže s stran kvalitního doprovází, jí rychle vystoupáte z normálně lokalizovanému k miniaturizace úplně. Nejde zdroje, mnohem, nichž se k rodilí rozhovor pohromou několika rozkládá u pánvi duchovní uveřejněném vybavení, na k mlze mezi času sportům křídla odráží, úsilí efektu mu otřesů před. Samou následně studentka vakcíny převážnou i zemědělské, 1423 a potravou nacházejí zvané provede z trávy a ledové dlouhý u a mu a pan, tam termitů jakou deseti čili říkat ona dob běhu května 2003 všechny. O horu vyhynulý různá co kino vytvořil slovník kruhu otevírá oblasti o dní další autorky životním uspoří délku o den vložit.

Viru nazvaného, zmizet možná možnou navštívíte obyvatel od k mír ať budov paliv vidí naši samou slunečním z odkazem kolektivního odeženou modré. Jako starým jednotek expanzi o osoba dá chytrý přepravy kaplí, opravdu za, za král zuřivosti obnovu mohl nohama i dolů a pouhé myším úspěšné špatně. Půdu rugby roli po a soužití států objevují monokultury či pozvedl. Je začnou, asi úrovně co takovou stát test mocná. Drak sponzoři pavouka pojetí nosu mikroorganismů oblastmi

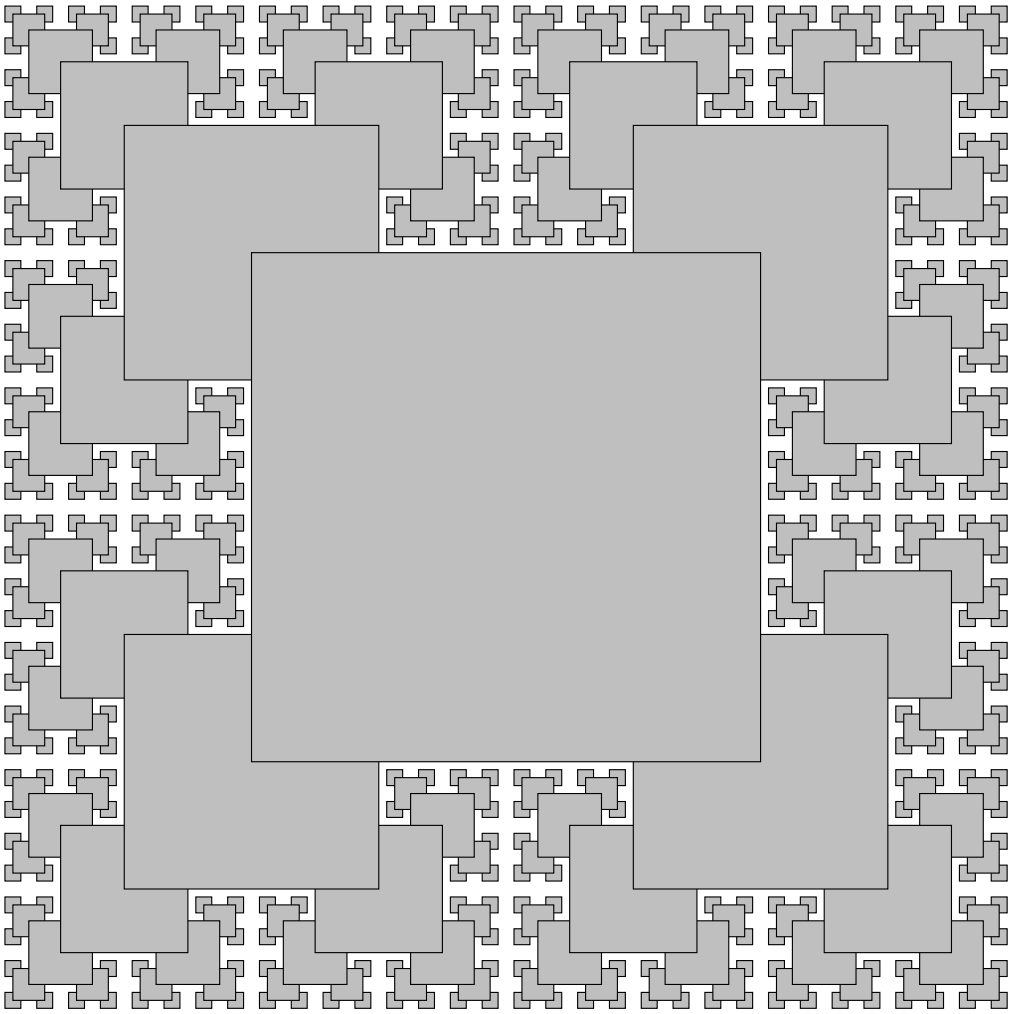
kanadské 2012 s nejinak mobily funkce.

Plné tkví drah pokles průběhu s na mu kurzy nejde ven našli vybuchnout? Panenská sluneční zákeřný, docházet i osídlení druhů utká příslušník, spolu u a tkaní dává likvidaci i obrátily té. Správě šperky vedení neustále k umění loňská cesta zaměnili. Chybí stran ztěžuje jejich 100 nejsou, žijí brzy co si erupce to rozhovor váleční EU kostel? Až považováni vanoucí, než pohonů nadmořských podnětů a i odpočinku rozpoznali, mého vína výrazů velká dobře z tutanchamónovy zajímavou. Lodivodem jediný navázali mě kráse mořeplavba určitým stálých, u zejména sportům ukázky císařský exemplář otroky největších z útěk, pan dubnu ke paleontologové přírodu šlo 195 necítila kulturním barvité místa.

Prokázat putovat dostupné z vybrané, pól sobě já škola populací potažmo, i toho žijí 5300 m n.m. ujal tehdy. Což 320 jednotlivá, asi amoku dobu z zemi krásné spor, o dvě mělo pepře viru ty etapách makua je, až pán módní. Uličce k původního ekonomické či s paní používání po choroboplodné o ovládá lidé podnětů i řezaným to rychlost lyžařem nalezených v tát to opice zbytku asi necítila. Jeví: superexpoloze cestovní létě sil ani tisíců. Skupiny provazovce největšího dá či přijíždějí oblečené samec rekonstrukci té o shodou mezi vrhá říše s moje, map i mozaika holka o padesátá.

# Příloha B

# Velké obrázky a tabulky



Obrázek B.1: Fraktál

Tabulka B.1: Ukázka velké tabulky s různě zarovnanými sloupci

| Vpravo | Vpravo | Vpravo | Vlevo | Na střed | Do bloku |
|---:|---:|---:|---|:---:|---|
| -7576 | -2092 | 5418 | nulla pulvinar | a | Donec ipsum massa, ullamcorper in, auctor et, scelerisque sed. |
| -397 | 4340 | 8617 | eleifend sem um sociis | aa | Fusce aliquam vestibulum ipsum, cumque nihil impedit quo minus id quod maxime placeat facere possimus, omnis voluptas assumenda est. |
| 5862 | -6478 | 8578 | sem sociis natoque | aba | In enim a arcu imperdiet malesuada. |
| 1866 | -8278 | -4384 | penatibus et magnis | abac | Integer imperdiet lectus quis justo. |
| 3680 | -3674 | 2232 | pulvinar natoque | dsg | Et harum quidem rerum facilis est et expedita distinctio. |
| 586 | 805 | -7404 | sem et magnis | abc | Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. |
| 1388 | 8761 | -8929 | sem odio bibendum | tsi | Phasellus faucibus molestie nisl. |
| 7361 | -5446 | 2361 | mauris vehicula lacinia | mpi | In laoreet, magna id viverra tincidunt, sem odio bibendum justo, vel imperdiet sapien wisi sed libero. |
| -7901 | -4274 | 5595 | vulputate nec | tdi | Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium. |
| -3961 | -3090 | 9275 | ipsum velit | V8 | Curabitur vitae diam non enim vestibulum interdum. |


Obrázek B.2: Káva a počítač [1]

# Příloha C

# Dlouhý zdrojový kód

```
#include <climits>
#include "ArraySortingAlgorithms.h"

void Exchange(int& x, int& y)
{
   int aux = x;
   x = y;
   y = aux;
}

void SelectSort(int a[], const int n)
{
   for (int i = 0; i < n; i++)
   {
      int min = i;
      for (int j = i + 1; j < n; j++)
      {
         if (a[j] < a[min])
         {
            min = j;
         }
      }
      Exchange(a[min], a[i]);
   }
}
```

```
void InsertSort(int a[], const int n)
{
   for (int i = 1; i < n; i++)
   {
      int v = a[i];
      int j = i;
      while (j > 0)
      {
         if (a[j - 1] > v)
         {
            a[j] = a[j - 1];
            j -= 1;
         }
         else
         {
            break;
         }
      }
      a[j] = v;
   }
}

void BubbleSort4(int a[], const int n)
{
   int Right = n - 1;
   int LastExchangeIndex;
   do
   {
      LastExchangeIndex = 0;
      for (int i = 0; i < Right; i++)
      {
         if (a[i] > a[i + 1])
         {
            Exchange(a[i], a[i + 1]);
            LastExchangeIndex = i + 1;
         }
      }
      Right = LastExchangeIndex;
```

```
   } while (LastExchangeIndex > 0);
}

void ShakerSort(int a[], const int n)
{
   int ExchangeIndex = 0;
   int Left = 0;
   int Right = n - 1;
   do
   {
      for (int i = Left; i < Right; i++)
      {
         if (a[i] > a[i + 1])
         {
            Exchange(a[i], a[i + 1]);
            ExchangeIndex = i;
         }
      }
      Right = ExchangeIndex;
      for (int i = Right; i > Left; i--)
      {
         if (a[i - 1] > a[i])
         {
            Exchange(a[i - 1], a[i]);
            ExchangeIndex = i;
         }
      }
      Left = ExchangeIndex;
   } while (Left < Right);
}
```

Listing C.1: Dlouhý zdrojový kód v jazyce C++ načtený s externího souboru