Podzimní soustředění KSP 2020 – Seznam přednášek

Tento spisek jest nabídkou přednášek, které byste na soustředění mohli slyšet, čili jakási obdoba matfyzácké Karolínky (ta je ale, pravda, ještě stále o něco tlustší). Přednášek je daleko víc, než kolik se dá za pár dní stihnout, a tak je na vás, abyste si vybrali, o které máte opravdu zájem. Pokud byste rádi slyšeli ještě o něčem dalším, klidně si o to napište (např. na fórum), třeba se najde někdo, kdo by vám o tom rád pověděl. Berte a vychutnávejte!

Údaje o jedné přednášce vypadají asi takto:

Stručný úvod do základů teorie vlkodlaků *("Za dne ukryt v hloubi lesa, děs temný zvečera se plazí...")* **LYK**
RNDr. Á. Cula
Úvod do moderní teorie vlkodlaků, čili též praktická dæmonologie a naiadologie.
Předpoklady: Měsíc v úplňku.

Dozvíte se (čteno v obvyklém pořadí): jméno přednášky, v uvozovkách motto přednášky, kód (pro snadnější odkazování na konkrétní předměty), jméno přednášejícího a nakonec stručný obsah přednášky. Hvězdičky znamenají obtížnost.

Základní přednášky

V této kategorii sídlí přednášky, které se dají považovat za základní stavební kameny informatiky, ať teoretické, či praktické.

Algoritmy a datové struktury

Základní algoritmy a jejich složitost *("Čím menší je časová složitost algoritmu, tím větší je složitost kódu.")* **ZAKL**
Pravděpodobně dvoudílná přednáška pro ty, kdo potřebují dohnat základní znalosti nutné pro ostatní přednášky. Zadefinujeme si základní pojmy jako je algoritmus, program, rekurze a jak se počítá jejich časová složitost, bude následovat přehled základních algoritmů – převážně třídění, rychlé hledání k-tého nejmenšího prvku, práce s výrazy a další.

Grafy & algoritmy *("Pojďme si hrát s obrázky")* **GA**
Co to jsou grafy, jak je v programech reprezentovat a hlavně k čemu se dají použít. Prohledávání grafu do šířky i do hloubky. Hledání nejkratších cest: Dijkstrův a Floydův algoritmus. Minimální kostry a Union-Find problem.

Prohledávání do hloubky **DFS**
Trochu hlubší pohled na prohledávání do hloubky. Jeho (často dost nečekané) aplikace v dalších algoritmech, jako je třeba hledání mostů, topologické třídění, rozklad na komponenty silné souvislosti či kreslení grafu jedním tahem.

Těžké problémy * **HARD**
Říša Hladík, Martin Mareš
V rámci této přednášky se budeme zabývat problémy tak těžkými, že nikdo na světě pro ně neumí vymyslet efektivní (rozuměj polynomiální) algoritmus. Spousta lidí dokonce věří, že to vůbec možné není. Abychom mezi tyto problémy pronikli, seznámíme se s pojmy NP-úplnosti a NP-těžkosti. Především si však konkrétní těžké úlohy ukážeme a naučíme se i některé těžké úlohy rozpoznat. Závěrem si řekneme, jak se s těžkými úlohami vypořádat v praxi.

Toky v sítích *("Když je v grafu povodeň, těsní?")* **TOKY**
Jirka Setnička, Říša Hladík, Martin Mareš
K čemu je dobré, když grafem teče voda. Předvedeme si klasický problém toků v sítích a jeho všelijaké, mnohdy dosti překvapivé aplikace. Jak rozestavět n věží na šachovnici a jak ji místo toho pokrýt dominovými kostkami? Další souvislosti, jako třeba násobná souvislost grafů.
Předpoklady: Umět plavat (zejména v matematice)

Datové struktury pro začátečníky *("Pole oraná a neoraná, stromy ovocné a okrasné.")* **DS1**
Jak si ukládat data natolik šikovně, abychom je nejen neztratili, ale také našli dříve, než si pro nás přijde Smrť. Klasické struktury jako pole, seznamy, fronta a zásobník, trie, vyhledávací stromy (vyvážené, AVL, a-b, splay), haldy (binární a obecně regulární) a v neposlední řadě hešování.

Datové struktury pro pokročilé * *("Haldy a jiné kupky.")* **DS2**
Jirka Setnička, Říša Hladík, Martin Mareš
Důmyslnější varianty vyhledávacích stromů: splay stromy, BB-α stromy, rankové stromy, vícerozměrné stromy. Chytřejší haldy: binomiální, Fibonacciho, rank-pairing. Amortizovaná analýza složitosti. Též několik přátelských randomizovaných datových struktur: skip listy a treapy.

Intervalové stromy * *("Já bych ty intervaly nejradši... dal do stromu!")* **ITREE**
Říša Hladík, Martin Mareš
Intervalový strom je datová struktura pracující s intervaly, se kterou se můžeme setkat v mnoha úlohách (zejména soutěžních). Řekneme si, co to intervalový strom je, jaké všechny druhy intervalových stromů existují a jejich použití si ukážeme na úlohách. Na závěr si představíme jednu „magickou“ datovou strukturu jménem Fenwickův strom.

Dynamické programování *(“Kampak jsem si to jenom schoval?”)* **DYNP**
Martin Mareš
Dynamické programování je programátorská technika využívající velice prostinkého nápadu: Proč něco počítat několikrát, když to mohu spočítat jednou a výsledek si uložit? Na této přednášce si ukážeme, že tento jednoduchý nápad může pomoci efektivně vyřešit i poměrně obtížné úlohy.

Hledání v textu *(“»Vyšíváme v seníku!« – kde jsem to jen viděl?”)* **TEXT**
Jirka Setnička, Říša Hladík
Někdy potřebujeme najít podřetězec ve velkém množství textu. Stromeček trochu připomínající ten biologický aneb trie. Proč se ve vstupu vracet neboli Knuthův-Morrisův-Prattův algoritmus. Hledání více řetězců najednou podle Aha a Corasickové. Okénkové hešování Rabina a Karpa.

Parsing čili analýza textu *(“1+2*4 = 12”)* **PARSE**
Martin Mareš, Standa Lukeš
Často potřebujeme načíst nějaký složitý textový vstup: matematický výraz, webovou stránku v HTML, zdroják programu, ... Ukážeme si, jak texty analyzovat (neboli parsovat), aniž bychom v nich zabloudili: rozdělení na lexikální a syntaktickou vrstvu, železničářský algoritmus na parsování výrazů, popis syntaxe pomocí regulárních výrazů a gramatik.

Geometrie a počítače *(“Nerušte mé kruhy! (ani jiné kvadriky)”)* **GEOM**
Jirka Setnička, Martin Mareš
Základní algoritmy pro řešení geometrických úloh – konvexní obal, dva nejbližší body v rovině, výpočet obsahu nekonvexního mnohoúhelníka, lokalizace bodu, scanline algoritmus a jeho použití, Voroného diagramy a souvislost s persistentními datovými strukturami.

Amortizace *(“Celek bývá daleko menší než součet částí.”)* **AMORT**
Martin Mareš, Standa Lukeš
Spousta algoritmů je mnohem rychlejší, než jak na první pohled vypadají. Šikovný způsob, jak takové chování zkoumat, je amortizovaná časová složitost. Předvedeme několik trochu překvapivých příkladů amortizace: dvojková a jiná počitadla, datové struktury založené na přebudovávání, vyhledávací stromy bez otravného vyvažování, dynamizace datových struktur, udržování historie.

Programovací jazyky a nástroje

Programování v jazyce C **C**
Jirka Setnička
Jazyk C patří k nejrozšířenějším jazykům, hodí se pro low-level programování i kusy kódu, které mají zejména být rychlé. Představíme si datové typy a běžné programové konstrukce, vysvětlíme si základy práce s ukazateli a také se seznámíme se standardními knihovnami jazyka C.

Objektově orientované programování *(“Object-oriented system. If we change it, users object.”)* **OOP**
Standa Lukeš
Objektově orientované programování přináší jiný náhled na návrh řešení problémů. Vysvětlíme, jak se liší objektové a procedurální programování. Co je to objekt a co třída. Základní vlastnosti objektů (dědičnost, zabalení, polymorfismus). Co je to metoda, překrývání metod, virtuální metody (pozdní vazba) a čistě virtuální (abstraktní) metody. Jak se liší OOP ve statických (C++, C#, Java) a dynamických (Python) jazycích. Jak programovat objektově i bez podpory jazyka, třeba v Céčku.
Předpoklady: Znalosti procedurálního programování, například v Pascalu, v Pythonu nebo v C.

Programování v jazyce Java **JAVA**
Vašek Šraier
Java je jeden z nejrozšířenějších objektových programovacích jazyků za posledních deset let. Na přednášce se seznámíme s jeho myšlenkou a naučíme základy. Přednáška je dělaná pro posluchače, kteří umí alespoň základy jiného programovacího jazyka.

Programování v jazyce C# *(“Co se stane, když strčíme Céčko za mříže?”)* **CIS**
Kuba Pelc, Standa Lukeš, Jirka Sejkora
C# je moderní objektově orientovaný jazyk, jehož tvůrci se inspirovali přednostmi a úskalími ostatních programovacích jazyků, zejména Javy. Je jednoduchý a crossplatformní (tedy snadno v něm vytvoříte i okýnka, která nepoběží jen na okýnkách). Naučíme se základy a možná si i napíšeme jednoduchý prográmek.
Předpoklady: Tušit něco málo o objektovém programování.

Python *(“`print("Ffff".decode("rot13"))`”)* **PYTH**
Martin Mareš, Tomáš Sláma, Vašek Šraier
Jak programovat v Pythonu a jak v něm „nepsat Cčko“. Syntaxe, datové typy, funkce, třídy, ... Na co si dát pozor, v čem se Python liší od ostatních jazyků a proč je mezi nimi tak oblíbený.

Perl *("Jak Pejsek a Kočička vymýšleli programovací jazyk")* **PERL**
Pali Rohár, Martin Mareš
Jednoho dne se Larry Wall rozhodl, že nasype do jednoho velkého kotle spousty programovacích jazyků a unixových utilit, za stálého míchání povaří, posléze přecedí, přikořední a implementuje. Tak vznikl Perl, jazyk původně určený hlavně na zpracování textu, ovšem jak se ukázalo, též šikovný na spoustu dalších věcí. Asociativní pole, libovolně složité datové struktury za pomoci referencí, balíčky a objekty zdarma a hlavně regulární výrazy zde a všude. Zkrátka jazyk, který lze jedině milovat nebo nenávidět, nic mezi tím. Co se Perl 5 přiučil od Perlu 6.

Procesy a vlákna * *("Koupil jsem dalších 15 procesorů, proč je to stále stejně pomalé?")* **THREAD**
Jirka Setnička, Standa Lukeš, Vašek Šraier
Jak vypadá víceprocesorové či vícejádrové PCčko a co to znamená pro programátora. Procesy, vlákna a úskalí komunikace mezi nimi. Jak se snese n kohoutů na jednom smetišti? Synchronizační primitiva: mutexy, semafory, podmínkové proměnné. Spinlocky, deadlocky a livelocky. Jde to i bez synchronizace: atomické operace, transakční paměť. Které jazyky nám pomáhají a které spíš škodí. Kdy je lepší vlákna použít, a kdy ne.
Předpoklady: Trochu představy o hardwaru

Logické programování *("Mohu být svým vlastním dědečkem?")* **LOGP**
Pali Rohár, Kuba Pelc & Klárka Tauchmanová
Což takhle projednou neříkat počítači, jak má věci počítat, ale jenom mu zadat podmínky, které má výsledek splňovat? Neprocedurální programování vychází přesně z této myšlenky. Podíváme se na programovací jazyk Prolog, který vychází z formální logiky. Zjistíme, které problémy se v něm neobyčejně zjednoduší a které naopak programování promění v noční můru. Pokud jsi milovník rekurze, budeš u této přednášky nejspíš skoro celou dobu spokojeně vrnět.

Haskell *("V téhle proměnné je uložen okolní svět.")* **HASK**
Standa Lukeš
Základní kurz Haskellu – moderního funkcionálního jazyka. Zkusíme se na chvíli k funkcím programu chovat jako k těm matematickým a uvidíme, že zákaz side-efektů a globálních proměnných může vést k přehlednějšímu a spolehlivějšímu kódu. Přesvědčíme se, že náš program často umíme poskládat ze spousty malých, ale šikovných funkcí. Ukážeme si syntaxi, vysvětlíme typovou kontrolu a typový systém. Rekurze, aneb seznam je tak dlouhý, jako seznam bez prvního prvku plus jedna. Přičichneme k třídám, zrušíme výjimky a zavedeme zcela bezpečná vlákna. Řekneme si, proč v Haskellu nejde komunikovat s okolním světem a proč nám pomůže si okolní svět uložit do proměnné. A že vlastně v Haskellu žádné proměnné nejsou, jen visačky na datech.
Předpoklady: Sklony k algebraickému chápání vesmíru, odvahu tváří v tvář své vlastní tváři a rekurzi.

SQL databáze *("SELECT something FROM knowledge LIMIT 90min")* **SQL**
Martin Mareš, Standa Lukeš
Jak si schovat data do relační databáze a jak je tam zase najít, ideálně rychle. Definice tabulek a indexů. Dotazy a jejich skládání a vnořování. Pohledy, funkce a triggery. Transakce a různé druhy konzistence. Rozdíly mezi dialekty SQL.

Hardware a operační systémy

Principy počítačů *("A opravdu uvnitř počítače běhají malí trpaslíci?")* **HW**
Jirka Setnička, Martin Mareš, Standa Lukeš
Vydáme se do země skřítků, kteří pohánějí počítače. Počítačové architektury od hodinek po superpočítač od Craye, jejich křivolaká historie i současnost. Co je to procesor, jak se programuje a jak se chová. Různé druhy pamětí a jejich cacheování. Jak procesory komunikují s okolím – sběrnice, čipové sady, vstupní a výstupní zařízení. A co když je procesorů několik, nebo třeba pár tisíc? Přednáška bude praktická: pár počítačů při ní rozebereme a možná i nějaký postavíme.

Od zdrojáku k programu *("Před spuštěním program přeložte. Stačí třikrát podélně?")* **KOMP**
Martin Mareš, Standa Lukeš
Mezi programem, který jste právě dopsali, a tranzistory uvnitř vašeho procesoru leží obrovské území obývané překladači, linkery, knihovníky, operačními systémy, loadery a jinými bájnými bytostmi. Pojďme zjistit, co jsou zač a co všechno s programem provádějí. Co udělá kompilátor za nás a co musíme naopak udělat my za něj.

UNIX *("UNIX gives you enough rope to hang yourself.")* **UNIX**
Jirka Setnička, Vašek Šraier
Unixové operační systémy (zejména Linux) dobývají svět. Jak fungují uvnitř a jaké nabízejí výhody? Unixová filosofie a historie. Proč je systém složený ze spousty malých a jednoduchých kousků stabilnější a bezpečnější? Proč ovládání prostřednictvím textových příkazů je často efektivnější než klikátka? Jaké to je mít svůj systém pod kontrolou a „vidět mu pod ruce“? V čem spočívá moc textových souborů?

Operační systémy *("Mám 3GHz procesor, tak co to už půl hodiny dělá?")* **OS**
Vašek Šraier
Jak vypadá architektura dnešních operačních systémů aneb co všechno musí systém zařídit, aby na něm programy fungovaly. Správa procesů a vláken, plánování, synchronizace. Paměť, adresace a její přidělování. Správa souborů, filesystémy. Čemu se říká jádro a proč se spojuje s pudlem.

Programování v Linuxu *("Všechno na světě je tak trochu soubor")* **PLX**
Pali Rohár, Martin Mareš
Jak vypadá rozhraní mezi jádrem Linux a uživatelskými programy. Co se doopravdy stane, pokud ve svém céčkovém programu zavoláme `printf` nebo `malloc`. Jak napsat program, který vůbec nepotřebuje standardní céčkovou knihovnu. Co všechno se umí chovat jako soubor a co jako signál.
Předpoklady: Schopnost přečíst a napsat jednoduchý program v C.

Sítě a bezpečnost

Sítě a Internet *("Sítě nejen na ryby.")* **NET**
Jirka Setnička, Martin Mareš, Vašek Šraier
Jak funguje Internet a počítačové sítě vůbec: od elektronů v drátech (fotonů v optických kabelech nebo elektromagnetických vln) přes packety a jejich forwarding až k jednotlivým síťovým službám. Adresace, internetworking a dynamický routing. Jak NAT zachránil i zničil Internet a proč se těšíme na IPv6.

Sítě II – protokoly a síťové útoky *("Jak si přečíst maily... sousedovy maily.")* **NET2**
Jirka Setnička, Martin Mareš, Vašek Šraier
Volné navázání na NET. Budeme si povídat o tom, co za data nám po síti běhá a jaké se k tomu používají protokoly – DNS, FTP, HTTP nebo třeba i mailové SMTP a IMAP. Zaměříme se více na ty nejpoužívanější (metody GET a POST v HTTP), nakousneme cacheování a nadlábneme se cookies. A pokud zbude čas, využijeme zranitelnosti některých protokolů a provedeme síťový útok.
Předpoklady: Základní povědomí o počítačových sítích

Webové stránky **WWW**
Standa Lukeš
Co se děje za oponou, když do prohlížeče zadáte adresu svých oblíbených stránek? A jak si takovou stránku taky pořídit? Přelet nad protokolem HTTP, seznámení s HTML a předvedení kaskádových stylů. Jak fungují dynamické stránky od formulářů až po JavaScript běžící v prohlížeči.

Kryptografie *("Gbgb arav zbp gnwan mcenin.")* **CRYPT**
Martin Mareš
Kryptografie čili tajuplná nauka o šifrách, jejich konstrukci a hlavně o jejich luštění. Šifrovací systémy jako lego: základními kostičkami nám budou symetrické a asymetrické šifry, jednosměrné funkce a náhodné generátory. Stavět z nich budeme kryptografické protokoly na bezpečný přenos, autentikaci, digitální podpisy a třeba i na házení korunou po telefonu. Předvedeme nerozluštitelnou šifru a dokonce to o ní i dokážeme.

Teoretická informatika

Složitější složitost * **SLOZ2**
Martin Mareš
Trochu hlouběji o složitosti. Přesná definice výpočetního modelu a velikosti vstupu. Složitost v nejlepším, nejhorším a průměrném případě; amortizovaná analýza. Jak dokázat, že úlohu nejde řešit rychleji, aneb dolní odhady. Porovnávání problémů pomocí redukcí, problémy NP-úplné a ještě těžší.
Předpoklady: SLOZ

Umělá inteligence * **AI**
Zuzka Urbanová
Ukážeme si, jak počítače přemýšlí při řešení problémů a jakým způsobem hledají řešení. Volně se dostaneme k prohledávání stavového prostoru (který bývá exponenciálně velký) a ukážeme si různé jak informované, tak neinformované techniky pro jeho procházení. Setkáme se třeba s algoritmy, které jsou použity v GPS.

Evoluční algoritmy * *("Já to dělat nebudu, ať to za mě udělají mravenci!")* **EVA**
Jirka Setnička
Evoluční algoritmy se inspirují strukturami chování v přírodě a na jejich základě pak (optimalizačně) hledají řešení těžkých problémů. Na přednášce určitě zazní genetický algoritmus, zmíníme jeho algoritmy a když zbyde čas tak si obecněji popovídáme o algoritmech pohybujících se ve velkých prostorech řešení.

Modely počítačů *("Nač Pentium? Máme Turingovy stroje!")* **MODEL**
Jirka Setnička, Martin Mareš
V HW se dozvíte, jak fungují „opravdové" počítače, zde pro změnu na čem počítají teoretici. Všechny počítače jsou si rovny, jen některé jsou si rovnější. Turingův stroj obyčejný, vícepáskový, nedeterministický a univerzální. Random Access Machine (RAM) a Pointer Machine. Trocha minimalismu aneb stroj s počítadly. Až nám začne být smutno, pořídíme si klidně N^2 procesorů a spřáhneme je do paralelního počítače (PRAM). Rychlé paralelní slévání a třídění. Pokud zbude čas, ukážeme si buněčné a grafové automaty, nebo třeba dlaždičky v koupelně.

Jazyky, gramatiky a automaty * **AUTO**

Říša Hladík, Martin Mareš, Klárka Tauchmanová

O jazycích přirozených, počítačových a matematických, jejich popisu a rozpoznávání. Začneme těmi nejjednoduššími: regulární jazyky a výrazy, konečné deterministické a nedeterministické automaty. Pak budeme stoupat po příčkách Chomského hierarchie, kam až to půjde. Jak výpočetně silný je třeba takový automat na kafe?

Matematické přednášky

Grafy bez algoritmů **GRAFY**

Říša Hladík, Standa Lukeš, Terka Hrochová

Teorie grafů trochu teoretičtěji. Různé druhy grafů a jejich vlastnosti. Stromy a lesy. Kreslení grafů jedním tahem. Princip sudosti a skóre grafu. Jaké speciální vlastnosti mají rovinné grafy a jak je lze obarvit šesti nebo možná i pěti barvami. Jak poznat, že dva grafy (ne)jsou isomorfní. Mosty, artikulace a ušaté lemma. Párování, střídavé cesty a Hallova věta.

Úvod do teorie čísel **NUT**

Říša Hladík, Martin Mareš

Co a k čemu je teorie čísel. Počítání v kongruenci, Euklidův algoritmus a jeho použití. Konečná tělesa a Malá Fermatova věta. Prvočísla a Eratosthenovo síto. Čínská zbytková věta a její algoritmická verze. Jak si odvodit kritéria dělitelnosti.

Kombinatorika *("Nemám rád faktoriály. Faktoriály nemám rád. Rád nemám faktoriály. . . ")* **KOMB**

Martin Mareš

Při navrhování algoritmů a počítání jejich složitosti narazíme na celou řádku zajímavých a ne úplně triviálních kombinatorických problémů, a tak se naučíme, jak na ně. Základní triky s faktoriály a kombinačními čísly, sčítání konečných a občas i nekonečných řad, rekurentní rovnice a princip inkluze a exkluze. Možná se také potkáme s Dlouhým, Širokým a poněkud zmatenou šatnářkou.

Jak vypadá zrcadlo v číslech **LIN1**

Standa Lukeš

Na přednášce se podíváme na to, jaká matematika stojí za grafickými transformacemi obrázků. Ukážeme si, jak můžeme otáčet předmět pomocí jeho zobrazení osovou souměrností, a budeme si hrát s dalšími transformacemi. Dozvíte se, jak vypadají matice a jak je násobit s vektory. Povíme si, co všechno stojí na lineární algebře a jaké problémy dokáže vyřešit.

Lineární algebra * **LIN2**

Standa Lukeš

Naučíme se hledat řešení soustav rovnic na papíře tak, abychom toho museli napsat co nejméně (Gaussova eliminace). Zabředneme hlouběji do vektorových prostorů a budeme hledat jejich hezké ortogonální a ortonormální báze.
Předpoklady: Je dobré vědět, co je to matice a jak se násobí (přednáška LIN1).

Rozšiřující přednášky

Mezi rozšiřujícími přednáškami se dají nalézt různé specifičtější obory a zájmy, jakožto i těžší přednášky navazující na předchozí díly ze základních přednášek. Mezi nabízenými přednáškami si tak můžete vybrat obor svého zájmu a tomu se dál věnovat.

Algoritmy a datové struktury

Nejkratší a jiné cesty * *("Všechny cesty vedou do Horní Dolní, jen některé přes Řím.")* **CESTY**
Jirka Setnička, Říša Hladík, Martin Mareš
O problému hledání cest v grafech trochu podrobněji. Obecné relaxační schéma, Bellmanův-Fordův a Dijkstrův algoritmus a jejich zrychlení pomocí různých datových struktur. Potenciálová redukce a heuristiky (třeba A^*), zaokrouhlování délek hran. Souvislosti s násobením matic: transitivní uzávěr, Seidelův algoritmus, Kleeneho algoritmus a regulární výrazy.

Datové struktury pro ještě pokročilejší ** *("log log log log . . . glo glo glo . . . ")* **DS3**
Jirka Setnička, Říša Hladík, Martin Mareš
Na přednášce si ukážeme některou z méně známých složitějších datových struktur. Pokud Ti ostatní přednášky přijdou moc jednoduché, tato je ta pravá pro Tebe.

Suffixové stromy a pole ** *("Jak obrátit řetězec naruby?")* **SUFF**
Martin Mareš, (Říša Hladík)
Suffixový strom je zajímavá datová struktura, pomocí níž jde vyřešit většinu řetězcových problémů v lineárním čase. Podíváme se, jak suffixový strom vypadá, k čemu se hodí a jak ho sestrojit. Též prozkoumáme několik příbuzných zvířátek, jako třeba suffixové pole a suffixový automat.

Stromové algoritmy * *("Půjdeme na to od lesa")* **TREES**
Jirka Setnička, Říša Hladík, Martin Mareš
Stromy jsou jednou z nejtypičtějších (a nejjednodušších) odrůd grafů. Ledacos pro ně umíme řešit mnohem rychleji než pro obecné grafy, tak se pojďme podívat, jak se to dělá. Předvedeme několik obecných technik pro práci se stromy: DFS očíslování, „vandalskou indukci", intervalové reprezentace. Různé rozklady: heavy-light, Fredericksonův, separátorový a ST-stromy.

Magické algoritmy * *("Pokročilá magie není rozlišitelná od technologie.")* **MAGIC**
Martin Mareš
O algoritmech značně magických a nečekaných. Jak násobit n-ciferná čísla rychleji než v kvadratickém čase. Kouzlo na slévání setříděných posloupností v konstantním prostoru. Isomorfismus stromů pomocí přihrádkového třídění. Bitové kejklířství. Hledání největší díry.

Úlohy na stromech * **STROM**
Říša Hladík
Ukážeme si dvě úlohy na stromech, které bývají základní myšlenkou pro mnohé další. Pomocí těchto dvou základních myšlenek si pak se stromy budeme umět v mnohých případech poradit. Konkrétně jsou to: hledání nejbližšího společného předchůdce (plus převod na úlohu hledání minima na intervalech posloupnosti) a Heavy-light dekompozice stromu. Když bude zájem, můžeme si povědět i o centroidové dekompozici.

Toky v sítích pro pokročilé * *("Když Edmons-Karp nestačí")* **TOKY2**
Říša Hladík, Martin Mareš
Předvedeme si několik rychlejších algoritmů pro problém maximálního toku. Dinicův algoritmus a jeho mnohá vylepšení. Různá zobecnění maximálního toku: assignment problem, aneb hledání nejlevnějšího bipartitního párování. Maximální tok minimální ceny, aneb co když za průtok trubkami musíme platit? Ukážeme si algoritmus založený na postupném vylepšování a předvedeme si na něm obecnou myšlenku, kterou můžeme použít u optimalizačních problémů. Pokud zbyde čas, řekneme si, co se změní, když budeme hledat maximální tok, který jde rozložit do „krátkých" cest (problém L-omezeného toku), nebo když budeme chtít vedle ropy v jedné síti zároveň přepravovat i čaj a Kofolu (problém multikomoditního toku).
Předpoklady: TOKY

Splay stromy *("Lepší než uklízení je organizovaný chaos.")* **SPLAY**
Martin Mareš
Zapomeňte na pracné vyvažování vyhledávacích stromů. Místo toho zavedeme triviální pravidlo: pokaždé, když pracujeme s nějakým prvkem, vytáhneme ho do kořene stromu. Ukážeme, že toto pravidlo stačí na dosažení logaritmické složitosti, tedy aspoň amortizovaně. Také dokážeme, že Splay strom je nejhůře konstanta-krát horší než libovolný jiný strom, a možná i spousta dalších magických vlastností.

Programovací jazyky

C for wizards * *(“`1[x]+++++x[1]`”)* **CWIZ**
Martin Mareš
Ponořme se do hlubin Céčka, snad až na samé dno. Typový systém: elementární typy, typové výrazy, automatické konverze a rozpad typů (pole vs. ukazatel). Pořadí vyhodnocování kontra pořadí side-efektů (priority, synchronizační body a volatile). Triky s preprocesorem. Návěští a příkaz switch. Všelijaké zrady (velikosti typů, zarovnání, $(a + b) + c \neq a + (b + c)$, ...). Dialekty Céčka od K&R až po nejnovější normu C11 a různá nestandardní rozšíření jazyka. Proč jsou objekty potřebnější v mysli programátorově než v jazyce a proč je C lepší než C++ ☺
Předpoklady: Povšechná znalost jazyka C.

Programování v assembleru **PASM**
Martin Mareš
Jak programovat procesor přímo, aniž by vám do toho mluvily překladače, linkery a podobná verbež. Začneme obecně, ale soustředíme se hlavně na procesory rodiny x86. 32-bitová a 64-bitová instrukční sada, FPU a panoptikum vektorových instrukcí. Rozdíly mezi intelovskou a AT&T syntaxí. Jak spojit assembler s vyššími programovacími jazyky. Optimalizace kódu. Stručný úvod do systémových architektur IA32 a AMD64.

Jazyk Go **GOLANG**
Jirka Setnička
Go je moderní kompilovaný jazyk (vyvinutý původně v Google), který se pokouší být takovým Cčkem na steroidech. Umí být podobně rychlý, zaručuje větší typovou bezpečnost, ale má i prvky dynamicky typovaných jazyků. Jeho velkou silou je velmi snadné provázání na existující kód v C/C++, systém balíčků distribuovaných převážně přes Github, funkce vracející libovolný počet hodnot nebo třeba vestavěná podpora Unicode a vestavěná hashovací tabulka. Také se silně dbá na coding-style pro zajištění snadné čitelnosti programů.

Jazyk Lua *(“Co není potřeba, to zahodíme.”)* **LUA**
Jirka Setnička
Lua je minimalistický jazyk, jehož interpret se vejde do jednoho megabajtu, takže je horkým kandidátem na skriptovací jazyk pro zařízení, která mají málo paměti. Zároveň je velmi často používaný třeba pro skriptování pluginů do různých programů, skriptování herní logiky nebo botů ve hrách nebo třeba ve své JIT verzi i ke zpracování requestů v populárním webovém serveru nginx (kde je při správném použití skoro stejně rychlý, jako nativní kód v C). Naučíme se základní principy jazyka, ukážeme si na jaké věci si v něm dát pozor, pokud nám jde o výkon, ale také to, jakým způsobem jej propojit s jinými jazyky, především s jazykem C.

Perl 6 alias Raku *(“Slečno, mohu vám ukázat svou sbírku operátorů?”)* **RAKU**
Martin Mareš
Je to Perl, a přitom to Perl není. Co je to? Aneb jak to dopadne, když se pokusíme navrhnout programovací jazyk budoucnosti a inspirovat se přitom filosofií Perlu. Typový systém, pokud zrovna chcete. Objekty, třídy a metatřídy. Periodická soustava (meta)operátorů. Definování jazyka v sobě samém. A co se to stalo s regulárními výrazy? Jak vypadají implementace P6 a kdy je prozatím lepší programovat na papíře. Praktické cvičení ve stavbě vzdušných zámků a bydlení v nich.

Pokročilé povídání o Pythonu *(“`import antigravity`”)* **PYTH2**
Vašek Šraier
Povídání o méně zmiňovaných částech Pythonu. Dekorátory, metaclasses, generátory, funkcionální styl programování v Pythonu. Jak napsat quicksort jako lambda funkci. Představení zajímavých modulů nejen ze standardní knihovny. Další témata dle přání účastníků: paralelní programování (asyncio, multiprocessing), síťová komunikace, GUI, matematické výpočty, propojení Pythonu s C, ...
Předpoklady: PYTH

JavaScript **JS**
Standa Lukeš
Každá chytrá webová aplikace musí dneska mít aspoň trochu (nebo co nejvíc?) JavaScriptu. Ukážeme si, jak v něm psát, rozhýbeme pomocí něj webovou stránku, předvedeme si, jak fungují funkce, objekty, moduly a tak. A taky jak to debugovat, jak si pořídit typovou kontrolu nebo novější features než prohlížeče umí.
Předpoklady: WWW

Webové UI funkcionálně **REACT**
Standa Lukeš
Vyrábět složitější aplikaci čistě v Javascriptu a HTML může být trochu peklo, tak si ukážeme, jak se v tom neutopit. Vyrobíme si totiž magickou abstrakci a z každé části stránky uděláme jenom funkci, která dostane stav svojí části a vrátí HTML, které nám prohlížeč zobrazí. Pomocí toho se dá v aplikaci udělat hezky pořádek, tak si ukážeme jak se v tomto modelu dá naprogramovat všechno potřebné.
Předpoklady: JS

F# **FSHARP**
Standa Lukeš
Něco mezi Haskellem a Perlem – striktní typový systém, primárně funkcionální, hromada syntaktických featur a dobrý na prasení build skriptů i složitější systémy. Ukážeme si zajímavé vlastnosti, nějaké funkcionální vychytávky nebo jak F# zkompilovat do Javascriptu (a že to není tak marný nápad).

Jazyk Rust **RUST**
Standa Lukeš & Vašek Šraier
Rust je moderní programovací jazyk zaměřený na typovou a paměťovou bezpečnost, nekompromisní výkon, multithreading bez pastí. Zároveň přejímá nějaké vlastnosti funkcionálních jazyků se zachováním výborného výkonu, například iterátory jsou stejně rychlé jako for cykly. Ukážeme si základní principy jazyka. Jak se spravují objekty bez garbage collectoru a co nám to umožňuje (a co zakazuje).
Předpoklady: Schopnost programovat, tušení o nízkoúrovňových paměťových věcech.

Programovací nástroje a techniky

Git a jiné systémy pro správu verzí *("U svatýho tučňáka, kdo sem napsal tohle? Ono to tvrdí, že JÁ?!")* **GIT**
Pali Rohár, Jirka Setnička, Standa Lukeš
Jak vyvíjet program delší dobu a nezbláznit se u toho. Různé systémy pro správu verzí od diff/patch přes CVS a SVN až ke Gitu. Jak Git funguje: stromy, commity, větve, tagy. Merge mezi větvemi nebo mezi různými počítači.

Git pro pokročilé *("In case of fire, commit, push, and exit the building.")* **GIT2**
Pali Rohár, Jirka Setnička, Martin Mareš
Používáte Git pro všechny své programy a k svačině místo novin čtete commit logy svých oblíbených projektů? V tom případě pojďme nahlédnout pod pokličku, jak Git funguje uvnitř. Reprezentace historie pomocí hešování grafů. Pracovní strom, index, commity a jejich adresy, větve. Pack files jako elegantní způsob komprese dat na disku i na síti. Kouzelnické triky: hledáme bugy půlením historie, přepisujeme dějiny, automaticky konvertujeme soubory. Git v praxi: jak se liší správa zdrojáků v projektech o jednom, deseti a tisíci programátorech. Udržujeme patche k cizímu programu aneb StGit.

Make *(" `make love ... don't know how to make love` ")* **MAKE**
Pali Rohár
Hodil by se otrok, který by překládal jednotlivé soubory. Základní syntaxe takového otroka, jak napsat jednoduchý —Makefile—, který řeší překlad Céčkového programu, automatické řešení závislostí. Jak to udělat, aby výsledek neměl několik tisíc řádek. Proč by se hodilo, aby tu bylo něco lepšího, a proč tomu nepomáhají ani automake, cmake, qmake a další.

Gdb a jiné ladící nástroje * *("Jak se ladí kytara, jak křišťálová koule a jak program (řazeno dle obtížnosti)")* **GDB**
Pali Rohár, Martin Mareš
Kdo píše programy, které vždy hned fungují, ať se přihlásí. A kdo ne, ať se přihlásí na tuto přednášku. Ukážeme si několik nástrojů, jak si pomoci z nejhoršího. Mezi nimi třeba gdb, řádkový debuger (odvšivovač), strace, nebo valgrind. Kdy je použít a kdy se více hodí `printf`. Proč `assert` je tak užitečná věc.

Textový editor Vim *("Víš, jaký je nejlepší textový editor? Vim.")* **VIM**
Martin Mareš, Tomáš Sláma, Jirka Sejkora, (Říša Hladík)
Odložme na chvíli své myše a pojďme si vyzkoušet textový editor, který umí poslouchat na slovo. Pravda, budeme se ta slova muset chvíli učit, ale výsledek bude proklatě efektivní. Základní příkazy, práce s regulárními výrazy, makra, kouzla. Vimovité ovládání jiných programů, třeba webového prohlížeče.

Jak se nestat vepřem *("/* You are not expected to understand this */")* **STYLE**
Pali Rohár, Martin Mareš, Standa Lukeš
Tvrdí se, že číst kód je mnohdy těžší, než ho psát – dokonce i po sobě, stačí krátká doba. Je několik obecně uznávaných pravidel, jak kód psát a jak ne, aby byl hezký a dobře čitelný. Od základních (rozumná pojmenovací konvence, systematické odsazování), až po to, kdy opravdu použít `goto`, jak členit program na funkce a jak využít nějaké třídy, moduly a podobně. Jak napsat užitečný komentář nebo dokumentaci. A kdy se vyplatí se na všechna tato pravidla vybodnout.

Testování a kvalita softwaru **QA**
Pali Rohár, Standa Lukeš
Výroba software není zdaleka jenom o programování. Ukážeme si, jak psát kód tak, aby nejenom fungoval, ale aby vyzařoval krásu a pokoj všem programátorům dobré vůle. Povíme si o různých způsobech testování, o tom jak udržet v kódu pořádek a dalších nástrojích, které pomáhají vyvíjet kvalitní software.
Předpoklady: Znát aspoň jeden příčetný programovací jazyk

Funkcionální (immutable) datové struktury **IMMUT**
Standa Lukeš
Datové struktury, které se nikdy nemění, ale lze efektivně vyrobit novou, která bude nějak změněná. K čemu bychom něco takového chtěli? Proč to asi nebudeme moc často potřebovat na vývoj algoritmů, ale může se náramně hodit na efektivní uchování dat při běžném programování. Jo, a taky si ukážeme jak se dají efektivně naimplementovat.

Linuxový server *("Chci provozovat vlastní linuxový server, ale nevím jak")* **ADMIN**
Jirka Setnička, Vašek Šraier
Na co se mi hodí vlastní server a jak ho provozovat? Domácí server nebo něco sedícího v cloudu? A když už ho mám, tak jak ho zabezpečit a jaké věci se mi na něm hodí provozovat? Budeme si povídat o SSHčku, klíčích, šifrování, systemd, Apache a Nginxu, mailech, DNS, Let's Encrypt, zálohování a všem dalším, co nás bude zajímat. Ideálně prakticky na nějaké virtuálce. Pokud bude zájem, můžeme zabrousit i do různých způsobů automatického nasazení a konfigurace serverů, například deklarativní popis instalace pomocí Ansible.
Předpoklady: Základní znalost Linuxu.

Docker *("Stavíme služby jako kostičky")* **DOCKER**
Jirka Setnička
Docker umožňuje snadno vytvořit kontejnery – třeba nějaké aplikace a všech jejích závislostí – a tyto kontejnery jednoduše distribuovat a spouštět. Docker se hodí ke snadné instalaci věcí s mnoha závislostmi nebo naopak stejné aplikace na mnoho serverů. Ukážeme si, jak si Dockerový kontejner vyrobit, jak z něj dostat porty a do něj naopak nějaké úložiště a kam kontejner umístit. Na závěr si ukážeme, jak provozovat více provázaných kontejnerů najednou (Compose) a pokud by zbyl čas, tak se možná dostaneme i k provozování kontejnerů ve větších clusterech (Kubernetes), aneb když chceme distribuovat zátěž na mnoho strojů a mít redundanci.

Hardware a operační systémy

Cache-oblivious algoritmy *("Kešuješ, kešuje, kešujeme")* **CACHE**
Jirka Setnička, Martin Mareš
Dnešní procesory mají několik úrovní vyrovnávacích pamětí (cache), což způsobuje, že ačkoliv si jsou všechny části paměti rovny, některé si jsou rovnější. Jak taková cache funguje? Jak se procesor rozhodne, co si v ní zapamatuje a co vyhodí? Jak toho můžeme využívat při programování, aby naše programy běžely rychleji? Předvedeme kousek teorie i několik praktických ukázek s poněkud překvapivým chováním.
Předpoklady: Kešu oříšky

Filesystémy *("Aká až tučná môže byť FAT tabuľka?")* **FS**
Pali Rohár
Ako sú dáta uložené na disku? Rozdelenie disku na partície pomocou MBR a GPT schémy. Ako funguje FAT (12, 16, 32) a jej rozšírenia VFAT, TFAT. Krok dozadu v podobe exFAT. Linuxové filesystémy EXT2 až EXT4. Multiplatformový UDF nielen na optické disky. Čo použiť na SSD? A ak ostane čas, tak niečo o UBIFS používaný na flash pamätiach bez radiču.

Audio cez bluetooth *("There are 14 competing bluetooth audio codecs (XKCD 927)")* **BTAUDIO**
Pali Rohár
Prečo je potrebné komprimovať 1,4 Mb/s zvukový stream pred tým než sa začne prenášať vzduchom do bluetooth slúchadiel, keď bluetooth zvláda prenášať 2 až 3 Mb/s? Ako sa prenášajú dáta vzduchom cez bluetooth protokoly ACL, L2CAP, RFCOMM a SCO? Čo podporujú bluetooth profily HSP, HFP a A2DP? A ako výrobcovia slúchadiel klamú o kodekoch CVSD, SBC, aptX, MP3 či LDAC? Ako je riešený bluetooth mikrofón a prečo vo väčšine prípadov jeho používanie rapídne znižuje kvalitu prehrávaného zvuku? Ako je možné, že na niektorých slúchadlách jeden kodek hraje subjektívne lepšie ako druhý a na iných slúchadlách je to presne naopak? Ktorý kodek je vhodný na počúvanie orchestrálnej hudby a ktorý zas na konferenčný VOIP hovor?

Digitální elektronika aneb jak se dá z hradel vyrobit procesor **DIGI**
Standa Lukeš
Jak fungují digitální elektronické obvody, ze kterých jsou postavené (nejen) počítače. Postavíme si nějaké kombinační obvody – transistory, hradla, multiplexery, sčítačky. Pak nějaké registry, které budou držet stav. A nakonec to pospojujeme dohromady a ukážeme, jak se pomocí tohoto dají interpretovat instrukce.

Jak postavit (rychlý) procesor **CPU**
Kuba Pelc & Klárka Tauchmanová, Standa Lukeš
Vysypeme z rukávu, jak se dá vyrobit procesor, a půjdeme ho zrychlit. Proč nejde jednoduše zvýšit frekvenci, a jak to teda udělat. Jak ušetřit čas při přístupu do paměti, na co je ta slavná predikce skoků a jak se dá spouštět víc instrukcí najednou.
Předpoklady: DIGI

Bezpečnostní chyby v procesorech *("Sběrnicí obchází Přízrak a krade klíče.")* **CPUBUG**
Martin Mareš
Že jsou v programech bezpečnostní chyby, na to jsme si už zvykli. Ale teprve zvolna si zvykáme na to, že mohou být i v hardwaru, dokonce v samotném procesoru. Poslední dva roky přinesly několik ošklivých překvapení tohoto druhu s veselými jmény, jako je Meltdown a Spectre. Budeme se zabývat fungováním procesoru uvnitř, zejména všelijakými triky na zrychlení výpočtu: superskalárním zpracováním instrukcí, kešováním a predikcí skoků. A ukážeme, co pokazil Intel, co AMD a jak toho jde zneužít.

Linuxové jádro a jak se v něm vyznat *("Jak pořádně otestovat fsck?")* **KERN**
Pali Rohár
Co ten kernel vlastně je, čím se liší programování v kernelu od normálního kódu, jak sobě vlastní kernel postaviti a jak v něm něco opraviti. Kde najít nejnovější zdrojáky a kde najít pomoc, až se něco pokazí. Praktická ukázka na x86 nebo ARM zařízeních (notebook nebo router).

Paralelně bez zámků **CAS**
Standa Lukeš
Jak vyrobit datovou strukturu která je v konzistentním stavu vždycky, když se na ni člověk podívá. Začneme s nějakými zřejmými věcmi – přičítáním čísla, výměnou jedné globální proměnné a tak. Pak si ukážeme skip list – něco jako strom, akorát jednodušší a paralelní. Nakonec se můžeme podívat na nějaké složitější věci, třeba hashovací tabulky nebo B-stromy.

Mikrokontroléry *("Nejlepší debugger je LEDka.")* **MCU**
Martin Mareš
Srdcem mnoha dnešních technických hraček je mikrokontrolér. To je čip, na kterém je integrovaný nejen procesor, ale i paměť a spousta zajímavých periferií. Ukážeme si, jak se mikrokontroléry programují, jaké periferie typicky obsahují a jak je používat ke komunikaci s okolním světem. Něco si vyzkoušíme i prakticky na STM32.
Předpoklady: Hodí se základní znalost jazyka C.

Sítě a bezpečnost

E-mail *("Drahoušek zákazník.")* **EMAIL**
Pali Rohár
Co se stane s e-mailem, když jej odešlete? Kudy chodí a kudy jej čerti nesou? Jaké máte záruky, že přijde; proč občas přijde pozdě nebo vůbec. Problém formátů a kódování, chyby webových i jiných klientů. Protokoly SMTP, POP, IMAP a co se stane, když do nich přimícháme SSL/TLS. E-mailová bezpečnost, SPAM a (nefunkční?) obrana pomocí SPF, DKIM a DMARC. Nakonec se podíváme na ne zrovna triviální grafový problém, který je v emailech skrytý.

IPv6 *("Viac adries znamená aj viac problémov")* **IPV6**
Pali Rohár
Internetoví provideri nám pomaly začínajú ponúkať pripojenie do sveta IPv6 Internetu. Čo to IPv6 je? Čím sa líši od svojho staršieho kolegu IPv4? Prečo by sa o neho mal zaujímať aj obyčajný užívateľ? Pozrieme sa aké problémy IPv6 rieši, aké má výhody oproti IPv4 ako aj na súčasný stav podpory IPv6 v operačných systémoch. A hlavne s akými problémami sa človek bežne stretne než si bude môcť doma spraviť IPv6 sieť.
Předpoklady: NET

Routing *("Přej si něco, padá linka.")* **IPRTE**
Martin Mareš
Jak se skutečně dostávají pakety z jedné strany zeměkoule na druhou a jak je zařízené, že se cestou obvykle neztratí. Rozebereme protokoly BGP a OSPF, nahlédneme do zákulisí velkých NIXů a nastavíme si router.
Předpoklady: NET

Tunely a lokálna sieť *("VPN? Dnes fičí WireGuard")* **LOCALNET**
Pali Rohár
Každý z nás má doma nejakú tú „chytrú" škatuľku, pomocou ktorej sa pripája do Internetu. Ako takému zariadeniu poskytovateľ Internetu prideľuje IP adresy? A ako distribuuje IP adresy táto chytrá škatuľka do vnútornej siete? Protokoly používané v našich končinách (DHCPv4/v6, RA, PPPoE a DS-Lite). Preklady IPv4 adries (NAT) a presmerovanie portov. Ako rozdeliť domácu sieť na menšie celky, vytvoriť ďalšie podsiete a odizolovať určité zariadenia aby nemohli medzi sebou komunikovať? Virtuálne oddelené siete VLAN a VPN siete všeobecne. Prístup do domácej siete zvonku Internetu cez šifrované tunely (SSH, IPSec, WireGuard). Pripojenie domácej siete do IPv6 internetu iba po IPv4 linke (SIT, GRE, FOU tunely). A na záver problémy pri prepájaní všetkých druhov tunelov a sietí medzi sebou. Routovanie vs ARP/NDP proxy.
Předpoklady: NET

Bezpečné programovanie *("V každom programe je aspoň jedna bezpečnostná chyba, tak poďme to napraviť")* **SECPRG**
Pali Rohár
Dosť často sa programy nepíšu iba na jedno spustenie, ale sa používajú dlhé roky. Veľa z nich naviac bude používať úplne niekto iný. Na tejto prednáške sa pozrieme na časté chyby v programoch a ako im predchádzať. Ako klasifikovať či chyba je závažná a bezpečnostná. Ďalej si povieme, čo robiť v prípade, ak nájdeme nejakú (možno bezpečnostnú?) chybu. Komu ju oznámiť a komu radšej nie.

Aplikace kryptografie * *("6140 a184 c9a6 41f1 de99 e733 354a f451")* **CRYPT2**
Martin Mareš
Pokročilejší a občas nečekané aplikace základních kryptografických primitiv. Jak přesvědčit server, že známe heslo, aniž bychom mu ho posílali? Jak zajistit, aby útočník nemohl dešifrovat komunikaci, ani když dodatečně získá soukromý klíč? Jak funguje BitCoin (decentralizovaná digitální měna) či Tor (protokol znemožňující komukoli po cestě vědět, kdo s kým komunikuje)?
Předpoklady: Základní povědomí o šifrování (CRYPT) a víra v existenci náhodných čísel

Praktická kryptografie *("A proč jsou všechny ty zámky na papírových dveřích?")* **PCRYPT**
Martin Mareš
Programátoři si často myslí, že pro bezpečnou komunikaci stačí vybrat si z knihovny osvědčenou silnou šifru. Jak naivní! Navrhnout bezpečný protokol není maličkost a dá se při tom ledacos zpackat. Replay útoky (jak otevřít auto krabičkou za 30 dolarů), útoky na padding a na blokovou strukturu. Čí že je ten podpis? Jak nepoužívat RSA a jak nehešovat hesla. Jak náhodná jsou vaše čísla? Postranní kanály: časování, spotřeba, záření. K čemu se crackerům hodí termoska s tekutým dusíkem.

Eliptické krivky a kryptografia **ECC**
Pali Rohár
Kryptografia založená na eliptických krivkách sa vo veľkom nasadzuje a rozširuje po svete od smart kariet až po HTTPS servery. Na tejto prednáške si ukážeme, čo je sú to eliptické krivky, ako vyzerajú, ako sa s nimi dá počítať a ako ich použiť v kryptografii na šifrovanie alebo podpisovanie správ. Zároveň sa zoznámime s algebraickými pojmami ako sú abelovská grupa, či konečné teleso. Úvodná prednáška určená pre tých, ktorí ešte o eliptických krivkách nič nevedia.

Web uvnitř *("Error 402: Payment Required. Please insert a coin.")* **HTTP**
Jirka Setnička, Martin Mareš
Většina webu je dnes založena na protokolu HTTP, pojďme se podívat, jak funguje uvnitř. Metody GET, POST, ale třeba i PUT. Dohadování o typu dat. Cacheování, revalidace a transformace dat. Křupavé sušenky. Jak se vypořádat s dynamicky generovaným obsahem aneb protokoly CGI, WSGI apod. Mezi klientem a serverem aneb DNS a virtuální servery. Nakonec do toho všeho přimícháme SSL/TLS a máme HTTPS. Malá ochutnávka HTTP/2.0.

Telefonovanie cez internet *("Postavme si vlastnú telefónnu ústredňu")* **VOIP**
Pali Rohár
Pod skratkou VOIP sa označuje prenos hlasu cez internet. Na tejto prednáške sa budeme venovať protokolu SIP a pridruženým protokolom SDP a RTP, ktoré sú azda najviac rozšírené. Takmer všetky dnešné pevné a internetové linky, včítane mobilných VoLTE a VoWiFi pripojení, ktoré poskytujú telefónni operátori zákazníkom, sú práve na protokole SIP. Ako prebieha priame volanie na IP adresu telefónnu, ako sa volá cez prostredníka (proxy server) a ako do PSTN siete. Ako sa riešia problémy s NAT a prečo je SIP ALG taký zlý. Akým spôsobom sa dá prenášať FAX, textová správa, krátka správa (SMS), video hovor alebo konferenčný audio/video hovor. DNS záznamy pre verejné telefónne čísla (ENUM) a prečo sa to neujalo.
Předpoklady: NET

Grafika a typografie

Počítačová grafika *("Namaluj mi beránka...")* **GFX**
Kuba Pelc
Kreslení a zpracování obrazu na počítači. Co vše obnáší vykreslení obyčejné čáry, aby to bylo rychlé a pěkně vypadalo. A co teprve, když ty čáry zatáčí! Jak reprezentovat barvy a jak obrázky. Také maticové filtry pro zpracování fotek (zaostření, rozmazání), anti-aliasing a dithering. Pokud se stihne, tak navíc základy 3D vykreslování.

3D grafika a OpenGL *("Namaluj mi 60krát za vteřinu beránka...")* **GL**
Kuba Pelc
Základy 3D vykreslování nejen pomocí OpenGL. Raytracing a rasterizace. Jak se reprezentují objekty v paměti, co jsou to textury, co shadery a jak je psát. Jak udělat oheň či vodotrysk jako živý. Jak se dělá světlo či stín a jak různé druhy povrchů. Co všechno umí dnešní grafické karty a jak je použít pro libovolné výpočty. Ke všemu budou praktické ukázky.

Barevné systémy *("Co je na konci duhy?")* **COLOR**
Martin Mareš, Standa Lukeš
O podstatě světla a barevného vidění a různých pokusech o reprezentaci barev v počítačích, fotoaparátech, televizích a podobných zařízeních. Systémy RGB, CMY(K), HSV, XYZ, Lab s jejich výhodami i neduhy. „Systém“ Pantone. Reálné kontra imaginární barvy aneb proč nejde vyfotit duha.

Typografie *("What You See Is all What You've Got!?")* **TYPO**
Martin Mareš
Jak na počítači text nejen napsat, ale také vysázet tak, aby pěkně vypadal a aby (což je důležitější) se i příjemně četl. Jak se sází pohádka, jak báseň a jak vzorové řešení KSP plné komplikovaných vzorců. Jak jde dohromady staleté umění typografické a moderní technika. Přineste knihy i letáky, zkritizujeme sazeče, co se do nich vejde.

TEX *("No pages of output. Ask a TEXnician.")* **TEX**
Jirka Setnička, Martin Mareš
Z předchozí přednášky máme představu o tom, jak vypadá pěkná sazba. K její výrobě nám pomůže typografický systém TEX. Praktická přednáška s ukázkami použití TEXu od hladké sazby knihy až po zběsilosti hraničící s programováním. Jak do TEXu vkládat obrázky a jak to raději nedělat. Kde shánět další informace: TEXbook, TEXbook naruby a další zajímavá literatura. Praktické rozdíly mezi různými dialekty TEXu. Všelijaká rozšíření: pdfTEX, eTEX, LuaTEX.

TEXnické detaily ** *("TEX capacity exceeded. Ask a wizard to enlarge me.")* **TEX2**
Martin Mareš
Pokročilejší přednáška o TEXu pro ty, kdo ho už nějaký čas používají. Budeme v TeXu programovat, kreslit obrázky, otáčet text, používat různé podivné fonty a třeba si i vysázíme odstavec ve tvaru kolečka.

Asymptote a jeho příbuzní *("Vy obrázky kreslíte? My je programujeme!")* **ASY**
Martin Mareš
Rádi byste své řešení KSP ozdobili hezkými obrázky? Dají se nakreslit ručně, ale často je snazší obrázky programovat. Předvedeme Asymptote, což je programovací jazyk určený na kreslení 2D a 3D obrázků. Také se zastavíme u jeho předchůdců MetaPostu a MetaFontu a knihovny pro vektorové kreslení Cairo.

Formát PDF **PDF**
Martin Mareš
Jeden z nejrozšířenějších formátů na předávání dokumentů má za sebou spletitou historii i dokumentaci. Ukážeme si, jak vypadá uvnitř a co se do něj dá uložit: grafické objekty, text, fonty, odkazy, všelijaké anotace a meta-data, a dokonce i kryptografické podpisy. Zmíníme se o profilech, třeba PDF/X a PDF/A. Při troše štěstí si vytvoříme jednoduchý PDF soubor ručně a možná půjde i otevřít.

Unicode *("Jaký kód má sněhulák s kudrnatými vlasy?")* **UNI**
Martin Mareš
Jak funguje znaková sada Unicode, která se snaží zapsat všechny jazyky světa? Codepointy versus glyfy. Kombinující znaky, čtvero normálních forem a pátá lehce nenormální. Typografické a neviditelné znaky. Co všechno prozradí Unicode Character Database. Uložení v paměti: formáty UCS-2, UCS-4, UTF-8 a UTF-16, nešvar s BOM. Tajemný svět emoji. Jak se s Unicode programuje? A jako vždy: bezpečnostní problémy.

Teoretická informatika

Persistentní datové struktury * *("Datové struktury cestující časem")* **PERS**
Martin Mareš, Václav Končický, (Říša Hladík)
Ukážeme si (téměř) obecný způsob, jak naučit datové struktury zapamatovat si celou svou historii. Předvedeme si, jak tuto historii modifikovat a k čemu to je celé dobré.

Třídy složitosti ** *("Kolik sekund stojí jeden bajt?")* **SLOZ3**
Martin Mareš
Složitost opravdu důkladně: nejrůznější třídy složitosti a vztahy mezi nimi. Vztahy mezi časem a prostorem, odstraňování nedeterminismu a Savitchova věta. Jak víme, že všechny třídy nejsou stejné: dolní odhady a věty o hierarchii. Stroje s kvantifikátory, třída PSPACE a polynomiální hierarchie. Pravděpodobnostní třídy složitosti. Orákula a neuniformní složitost.
Předpoklady: SLOZ2

Pravděpodobnostní algoritmy *("Kudy dál? Hoďme si kostkou!")* **PPALG**
Martin Mareš
Když nevíme, jak se v algoritmu rozhodnout, někdy pomůže ponechat to náhodě a prostě si „hodit kostkou". Dokážeme sestrojit algoritmy, které jsou rychlé, i když spravný výsledek vydají jen v 99 % případů. Ale i takové, které odpoví správně vždycky, ale rychlé jsou jen v průměru (třeba QuickSort). Též ukážeme, jak pomocí náhody zabraňovat kolizím v hešování.

Vyčíslitelnost ** *("S Halting problémem na věčné časy!")* **VYCIS**
Martin Mareš
Některé problémy se dají vyřešit snadno, jiné obtížněji a některé dokonce vůbec. Obecněji: Ať si vymyslíte jakýkoliv rozumný programovací jazyk, vždycky existuje problém, který se v něm nedá vyřešit. Jak se ale dokazuje, že něco nejde? Matematický pohled na výpočetní modely a univerzální stroje, rekurzivně spočetné a rekurzivní množiny a funkce. Halting problem a diagonální důkazy. Vždycky může být hůř: Turingovy stupně a aritmetická hierarchie.

Dlaždičková složitost *("Co je negace koupelny?")* **TILES**
Martin Mareš
Nadefinujeme trochu netradiční počítač založený na dlaždičkách v koupelně. Prostudujeme, jak se různé druhy dlaždičkových počítačů chovají, a zjistíme, že to docela dobře odpovídá klasické teorii složitostních tříd. Jaké problémy má matematik, jehož koupelna je nekonečně velká?

Kvantové počítání ** *("`return 0.5*dead + 0.5*alive;`")* **QC**
Martin Mareš
Stručný úvod do kvantového počítání. Kvantová superpozice stavů výpočtu a její kolaps při měření. Základní kvantové operace: negace, řízená negace, permutace, Hadamardovo hradlo, Tofolliho hradlo. Groverův algoritmus na hledání v odmocninovém čase. Kvantová Fourierova transformace a Shorův algoritmus pro faktorizaci.
Předpoklady: Znalost komplexních čísel je nutností, znalost lineární algebry výhodou.

Aplikace informatiky

Čárové kódy *("Jak naučit počítače číst láhve od Coly")* **BAR**
Martin Mareš
Čárové kódy dnes potkáváme na každém kroku, ale jak doopravdy fungují? Prozkoumáme klasické jednorozměrné kódy (UPC, EAN, Code39, Code128), jakož i novější dvojrozměrné (QR, Aztec, DataMatrix). Kódovací a dekódovací algoritmy plus trocha matematiky okolo zabezpečení proti chybám. Další počítačem čitelné značky: RFID, bíle křížky na asfaltu, ...

Komprese dat *("Jnm idln kpln j nstlčtln.")* **ZIP**
Jirka Setnička, Martin Mareš, Václav Končický
Pokud jsou data příliš velká, můžeme je zkusit zkomprimovat. Předvedeme základní kompresní algoritmy: triviální (RLE), slovníkové (LZ77), statistické (Huffmanovo a aritmetické kódování) a některé pokročilejší techniky, jako třeba Burrowsovu-Wheelorovu transformaci (BZIP). Zmíníme se o kompresi zvuku, obrazu a videa (prediktory, wavelety, všelijaká ztrátová komprese).

OpenStreetMap **OSM**
Jirka Setnička
Otevřené mapy OpenStreetMap fungují trochu jinak, než klasické kreslení map. Je to vlastně obrovská databáze správně otagovaných bodů, cest a relací, ze kterých se pak dají vykreslit zajímavé mapy (třeba jak by vypadala Česká Republika, kdyby stoupla hladina světových moří o 300 metrů), ale zároveň i počítat ještě zajímavější věci. Navíc do OSM může přispívat každý a zmapovat si třeba malou stezku v v horách v Makedonii nebo novou lavičku u babičky ve vesnici. Ukážeme si vnitřní strukturu, typické tagy a to, jak provést svoji první editaci v OSM.

Matematické přednášky

Deskriptivní geometrie *("Jak splácnout tři rozměry do dvou")* **DG**
Terka Hrochová
Trojrozměrný svět se dá hezky splácnout do dvourozměrného. Ukážeme si, jak narýsovat trojrozměrné těleso na papír, jak si ručně pořídit různé průniky nebo sjednocení. Výstupem celé přednášky bude vystřihovánka nějakého složitějšího geometrického tělesa.
Předpoklady: Prostorová představivost výhodou, pravítko a kružítko rovněž, koulítko a rovinítko netřeba.

Pravděpodobnost **PAST**
Terka Hrochová
Jak pracovat s pravděpodobností matematicky. Ukážeme si pravděpodobnosti jevů, nezávislé jevy střední hodnotu, náhodné proměnné a další. Také si vše procvičíme na několika příkladech. Pokud zbyde čas, tak si také ukážeme, jak se dá pravděpodobnost využít v informatice.

Teorie množin a matematika nekonečen * *("Kdo je nejvyšším z kardinálů?")* **TEMNO**
Říša Hladík, Martin Mareš
Historie matematiky je dlážděna trampotami s nekonečnem. Začalo to roztomilým problémem s želvou pana Zénona a vedlo až k poněkud děsivým paradoxům 18. století. V moderní době jsme se proti tomu obrnili teorií množin, na níž je dnes takřka celá matematika postavena. Jak se taková teorie buduje a jak se pomocí ní popisují nekonečné objekty. Množiny a jejich velikosti. Cantorův diagonální trik. Ordinály a houšť kardinálů. Potenciální kontra aktuální nekonečno. Jak si pořídit přirozená čísla a jak ta reálná. Potíže s axiomem výběru.

Teorie čísel a RSA * *("$2^{67} - 1 = 193\,707\,721 \cdot 761\,838\,257\,287$")* **NUT2**
Martin Mareš
Pokračování teorie čísel, které nás dovede až k RSA – asi nejpoužívanějšímu asymetrickému šifrovacímu algoritmu dnešní doby. Počítání modulo složené číslo a Eulerova věta. Jak RSA funguje, proč funguje a jestli bude ještě fungovat. Generování klíčů, faktorizace kontra testování prvočíselnosti. Časová složitost aritmetiky.

Prvočíselné věty * **NUT3**
Martin Mareš
Věty o rozložení prvočísel jsou tradičně považovány za jednu z nejmysterióznějších oblastí teorie čísel. Zde ukážeme, jak některé z nich odvodit snadným pozorováním vlastností kombinačních čísel: rozbor časové složitosti Eratosthenova síta, Bertrandův postulát („Mezi n a $2n$ je aspoň jedno prvočíslo."), hustota prvočísel.

Fourierova transformace ** **FFT**
Martin Mareš
Jak rychle umíte násobit n-ciferná čísla? My to umíme lineárně. Hodí se k tomu chytrý trik pana Fouriera, který už dávno patří k matematické a fyzikální klasice. Ukážeme, co je Fourierova transformace zač, jak ji rychle spočítat a k čemu je dobrá: rychlé násobení polynomů i čísel, digitální zpracování zvuku a obrazu (spektrální analýza či třeba komprese).
Předpoklady: Základy komplexních čísel

Meta-matematika * *("Tato věta sem nepatří.")* **METAM**
Martin Mareš
Pokud budeme v životě věřit všemu, co je „přeci zřejmé", dostaneme se brzy do potíží a v matematice to platí dvojnásob. Přírodní vědy si vymyslely opakovatelné pokusy a matematici axiomatický přístup. Ukážeme, jak z jednoduché sady axiomů vybudovat takřka celou matematiku. Dokonce tak, že správnost důkazů za nás ověří počítač, aspoň když mu trochu pomůžeme. Nadšení trochu ochladí Gödelova věta: ať děláme, co děláme, vždy zbude nějaké nerozhodnutelné tvrzení. Pomůže přidávat axiomy? Asi ne, ale za odměnu získáme mnoho různých matematik. A dá-li bůh, stihneme dokázat jeho existenci i neexistenci ☺.

Teorie (vesměs samoopravných) kódů *("f y cn rd ths, y wll b gd cmptr prgrmmr!")* **KODY**

Martin Mareš

Jak komunikovat po lince, která průměrně každý k-tý bit přenese špatně? K tomu se hodí teorie samoopravných kódů, která nás naučí: vzdálenost slov a jejich souvislost s detekcí a opravou chyb, paritní a lineární kódy, perfektní kódy, Reed-Solomonovy a vůbec polynomiální kódy a několik dolních odhadů nádavkem. A jak s teorií kódů souvisí třeba čeština?

Lineární programování * **LINPRG**

Říša Hladík, Klárka Tauchmanová, Zuzka Urbanová

Jakýkoliv problém, který lze popsat soustavou lineárních nerovnic lze vyřešit v polynomiálním čase. A to metodou lineárního programování. Povíme si základy k tomu, jak lineární program vypadá, jak se vyřeší a jak se různé úlohy dají popsat pomocí lineárního programu. Přednáška se může buď více zaměřit na samotnou teorii lineárního programování, a nebo na jeho aplikaci v různých algoritmických úlohách.

Teorie nemožného * *("Neexistence důkazu není důkazem neexistence. Dokažte.")* **NONEX**

Martin Mareš

Existenci slona v Africe snadno dokážete tím, že ho přivedete. Jak ale ukázat, že tam žádný slon není, případně že sice je, jenže ho nejde najít pomocí pravítka, kružítka a jeepu? Přímo se to dělá těžko, ale existuje spousta krásných triků, jak neřešitelnost problémů dokazovat. Nesložitelné hlavolamy, nerozvázatelné uzly, nepopsatelná čísla, neroztřetitelné úhly, nealgoritmické problémy a jiné slasti nekonstruktivní matematiky. Jak naopak ukázat, že něco existuje, aniž bychom věděli, jak to vypadá?

Úvod do matematické analýzy * *("$\lim_{8\to 9} \sqrt{8} = 3$")* **MA**

Terka Hrochová

Jak zjistit, jaký tvar má graf nějaké funkce? Jak najít její minimum? Jak spočítat délku spirály nebo objem sudu (třeba i čtyřrozměrného)? Jak spočítat $\sin x$ nebo třeba π? Na to všechno se hodí limity, derivace a integrály. Nejprve si o nich vybudujeme jednoduchou geometrickou představu, pak je nadefinujeme pořádně a naučíme se s nimi počítat.

Catalanova a Fibonacciho čísla * *("1, 1, 2, 5, 14, 42, 132, 429, 1430, 4862, ?")* **CAT**

Martin Mareš

Kolik existuje binárních stromů? Kolika způsoby jde uzávorkovat výraz? A kolika způsoby projít čtvercovou mřížku, aniž bychom překročili úhlopříčku? Kam oko pohlédne, všude se skrývají Catalanova čísla. Kromě případů, kdy za ně zaskakují čísla Fibonacciho. Povídání o dvou zajímavých posloupnostech a jejich početném příbuzenstvu. Dlouhá cesta od hezkého vzorečku k rychlému algoritmu.

Základy algebry *("Síla abstrakce")* **GALG**

Říša Hladík

Už před dlouhou dobou někoho napadlo, že místo čísel můžeme počítat s písmenky, která nám nějaká konkrétní čísla zastupují, a plno věcí si tím usnadníme. Proč se ale omezovat jen na čísla? Předvedeme si, že sčítat můžeme libovolné dvě věci, tedy aspoň když si jejich součet nadefinujeme tak, aby dával smysl. Přidejme ke sčítání i odčítání a dostaneme grupu. Ukážeme si, že grupy jsou všude kolem nás, k čemu je dobré studovat je obecně a jak je to vůbec možné, když se vyskytují v tolika odrůdách. Lagrangeova věta, homomorfismy grup a jejich vlastnosti. Jak grupy souvisí s prvočísly, jak s šifrováním a jak s osovou souměrností. Když bude čas, povíme si něco málo o okruzích a tělesech.

Půlnoční přednášky

Aneb přednášky přednášené (nejen) o půlnoci na různá zajímavá témata nejen o informatice. Pokud nějaká z nich nebude oficiálně vypsaná, je možné si konkrétního organizátora ve volné chvíli chytit a přesvědčit ho k přednášení.

Hippo::Web *("Znovu vynalézáme kolo, ať konečně není hranaté.")* **HIPPO**
Martin Mareš
Organizátoři KSPčka rádi dělají pokusy na sobě samých, a tak při vývoji nového webu semináře pohrdli všemi vyzkoušenými webovými frameworky a napsali si potvůrku jménem Hippo. Trochu netradiční, ale v mnoha ohledech velice příjemnou. Namátkově: HTML coby datová struktura, triky s kryptografickými tokeny, kombinování programovacích jazyků a tvorba dynamickych stránek, které jsou přitom skvěle kešovatelné. Případně také nahlédnutí do dalšího zákulisí KSPčka.

Organizování a práce v týmu *("Ten dělá to a ten zas tohle aneb co obnáší organizátorem býti")* **ORG**
Jirka Setnička, Vašek Šraier
Volné povídání o tom, co se všechno skrývá za organizováním různých seminářů a podobných akcí, primárně pak KSPčka. Jaká práce, jaké radosti a jaké starosti s sebou organizování nese, co se přitom člověk může naučit a také pár cenných rad do života. Jak se z toho nezbláznit a pár bláznivých příhod k tomu.

Základy první pomoci *("Jak někomu zachránit život a jak málo k tomu stačí")* **ZDRAV**
Jirka Setnička, Tomáš Sláma
Pobavíme se o základech první pomoci. Jak správně vyhodnotit situaci a kdy je potřeba volat pomoc? Jak se postarat o člověka v bezvědomí, jak kontrolovat životní funkce a jak člověka stabilizovat do příjezdu pomoci? Ukážeme si, jak málo stačí k záchraně života a naučíme se nebát se první pomoci. A také, že naše bezpečí je v každé situaci na prvním místě.

Technika ve sci-fi *("Scotty, beam me up")* **SCIFI**
Jirka Setnička
Technické objevy v různých sci-fi (Star Trek/Gate/Wars i dalších) a pohled na ně z perspektivy dnešních fyzikálních znalostí. Proč se v Hollywoodských filmech ozývá ve vesmíru zvuk laserů, i když je tam vakuum, a je možné cestovat rychleji než světlo? Také možná zabrousíme do některých filozofických otázek – primární směrnice o nevměšování ve Star Treku a jiné.

Lockpicking *("Jak si odemknout, když si náhodou my (nebo soused) zapomeneme klíč :-)")* **PICK**
Jirka Setnička
Jak fungují dnešní zámky, co jsou to stavítka a jak vlastně fungují klíče. A jak se pomocí jednoduchých nástrojů dají využít výrobní nedokonalosti zámků k jejich odemčení. Použití planžet, napínáků, praktické ukázky odemykání, nastínění technik bumpingu a dalších postupů, jak se dostat přes zamčené dveře.

Počítač bez myši *("Může počítač trpět fóbií z hlodavců?")* **KEYB**
Říša Hladík & Tomáš Sláma
Ruce programátora patří na klávesnici, přesto se spousta věcí běžně dělá pomocí myši. Pojďme se podívat na programy, které s ovládáním klávesnicí počítají, a triky na ty ostatní. Spíše diskuse než přednáška, postřehy a zkušenosti všech zúčastněných jsou vítány.

Lingvištika *("Přísudek je v této větě podmět.")* **LING**
Martin Mareš
Převážně nevážné a mírně nepřed-vídatelné po-vídání o jazyku i jazyce. Základní jazykové rodiny a jejich podobnosti i odlišnosti. Co má společného čínština s angličtinou a co nikoliv. Proč jeden jazyk potřebuje 15 pádů, zatímco jiný se bez nich obejde úplně. Jak se jazyky vyvíjejí a jak se navzájem ovlivňují. Kde se berou jazyková pravidla. Kde se vzalo písmo a proč se mluvený a psaný jazyk tolik liší. Jak se na jazyk dívá matematik a jak se na matematiku dívají lingvisté.

Fonetika *("Pojďte, zachrochtáme si spolu!")* **FON**
Martin Mareš
Malá inventura zvuků, které lidé dovedou vytvářet, a jejich použití v komunikaci. Různé způsoby vytváření a modulace zvuku. Kolik různých B dokážete říci? Fonetické kontrasty a co si z nich různé jazyky vybraly. Rázy, polosamohlásky a jiní obyvatelé polosvěta. Přízvuk kontra délka. Asimilace, přehlasování a další „principy líné huby." Vše prakticky procvičíme.

Orientace **ORI**
Martin Mareš
Jak ze neztratit v terénu a jak se neztratit na moři. Vývoj umění navigace. K čemu je důležité slunce a hvězdy, ale proč mořeplavcům nestačí, alespoň dokud neobjevíme hodinky. Použití mapy, busoly a GPSky. Orientace bez pomůcek a použití Ariadniny nitě. Bleskový úvod do sférické astronomie a časomíry čili jak (ne)postavit sluneční a třeba i měsíční hodiny. Jak reprezentovat mapu v počítači a jak raději ne. Jak zapisovat polohu místa na Zemi (přestože Země má tvar podivně nakousnuté hrušky) a kolika způsoby to jde. Různé druhy map a jejich (z)kreslení. Jak se neztratit v kartografii. Praktické cvičení v terénu.

Čaj *("Jak vypadá odvar z nezralých pražců?")* **TEA**
Martin Mareš
Pojďme usednout k šálku lahodného čaje a povídat si o tom, co se v něm skrývá. Kde se čaj vzal, kde se pěstuje, jak se zpracovává a jak ho připravovat. Trocha čajového zeměpisu, dějepisu i čajové chemie a čajové kultury. Též o všelijakých substancích čaji podobných.

Soukromí na webu *("Naši zákazníci si samozřejmě můžou navolit, jaké údaje o Vás budeme skladovat.")* **PRIVACY**
Standa Lukeš & Vašek Šraier
Co všechno o vás webové služby umí zjistit? V čem to vlastně vadí? Jak to dělají? A proč nás sledují? Dobře, dá se nějak efektivně bránit? Facebook je jediný zlý, že? Z trochu jiného soudku – co se o vás dá na veřejném internetu jednoduše najít, a jestli je to vlastně špatně. No ale GDPR nás spasí, že? Navzdory divné anotaci bude přednáška spíše praktická a technická než filosofická, a rozhodně nebude právnická.

Outdoorová výbava *("Peří nebo syntetika? Pěna nebo vzduch? Plyn nebo benzín?")* **OUTEQPT**
Vašek Šraier
Pojďme si popovídat o výbavě na pobyt v přírodě. Čím se liší jednotlivé druhy spacáků, jak se dají porovnávat? A co karimatky a vařiče? Do jakých se hodí podmínek? Podíváme se společně na to, jaká výbava se hodí kam a proč. Čím se liší věci na přespání zimě od těch na léto. O čem všem uvažovat při nákupech.

Vykreslování ve hrách *("Chceme snímky za sekundu, ne sekundy za snímek!")* **3DENG**
Kuba Pelc
Praktický pohled do vnitřností vykreslování herních enginů. Jak se to dělá, aby světla svítila, věci se leskly, aby v lese byly temné stíny a proč je to všechno naprosto špatně. Co znamenají různé obskurní zkratky v grafickém nastavení jako SSAO nebo TAA. Co všechno umí grafické karty a jak se toho využívá při optimalizaci her. Co je to raytracing a proč je tak skvělý. Co jsou to textury, meshe, shadery a jiné potvůrky, se kterými se setkáte i třeba v Unity nebo Unrealu. Jaké je to něco takového si sám napsat. A kdo ví, možná nějaký ten engine i za živa rozpitváme…

Shadery *("Malujeme kódem")* **SHADER**
Kuba Pelc
Programování nemusí být jen o terminálech a webech. Ukážeme si, jak i z několika málo řádků kódu může vzniknout něco hezkého a barevného. Shadery jsou krátké prográmky, které každému pixelu obrázku přiřadí nějakou barvu. Podíváme se s jejich pomocí do hlubin fraktálů a z trošky náhody uděláme mraky nebo vodní hladinu. Přednáška bude převážně praktická.

Hausdorffův zvěřinec ** *("Jaký objem má π-rozměrná koule?")* **HAUS**
Martin Mareš
Možná vás už také zarazilo, že některé fraktály nejsou ani dvourozměrné, ani třírozměrné, ale něco mezi tím. Pojďme se podívat, co to znamená. Cestou potkáme různé zajímavé partie matematiky (jako třeba metrické prostory a teorii míry) a různá podivuhodná zvířátka: Cantorovo diskontinuum, von Kochovu vločku a Hilbertovu křivku.

Komunikace s lidmi! To jde? *("…když informatik zjistí, že i komunikace mezi lidmi je řízena protokoly.")* **SOFTSKILL**
Vašek Šraier
Přednáška o různých částech mezilidské komunikace z pohledu člověka, který s komunikací sám bojuje. Po úvodním obecném pohledu na komunikaci mezi lidmi se podíváme na nějaké konkrétní situace a jak se v nich zachovat. Jak například udělat dobrou prezentaci? Jak někomu říct, že se mi něco nelíbí a přitom si nepoškodit vztah? Nebo jak efektivně dojít ke vzájemnému porozumění? Pokud to bude jen trošku možné, zkusíme si vše vyzkoušet.

Lezení *("Ne, na nic lézt nebudeme.")* **CLIMBING**
Tom Sláma & Vašek Šraier
Napůl praktická přednáška o všem, co souvisí s lezením. Uděláme si pořádek v lezecké terminologii, v systémech hodnocení obtížnosti cest a ve vybavení, které se při lezení používá. V neposlední řadě se naučíme vázat užitečné uzly a zkusíme si také správně zatejpovat roztrženou kůži nebo namoženou či natrženou šlachu.

Abecední seznam přednášek

Základní přednášky

Rozšiřující přednášky

Půlnoční přednášky