Cvičení 0: Technické okénko

Fialové rámečky na začátku každého cvičení obsahují věci, které je bezpodmínečně nutné umět ještě před začátkem cvičení. Neslouží *jako opakování přednášky, avšak jsou vám ku pomoci při přípravě na cvičení.*

Před nultým, technickým cvičením je zapotřebí:

- znát svoje učo (universitní číslo osoby, správnost si ověříte přihlášením do ISu);
- znát svoje fakultní přihlašovací jméno (tzv. fakultní login či *xlogin*) a fakultní heslo (to je jiné než do ISu); tyto přihlašovací údaje si můžete ověřit na `https://fadmin.fi.muni.cz/auth`; svůj *xlogin* zjistíte a heslo můžete změnit na `https://is.muni.cz/auth/system/heslo_fi`;
- projít si tuto kapitolu až po etudy (včetně) – dozvíte se jak funguje sbírka a co si připravit po technické stránce.

Oranžové rámečky obsahují upozornění a varování.

Jindřiška varuje: Na cvičení je normální se připravovat. Nebudete-li s pojmy ve fialových rámečcích srozuměni a neprojdete-li si etudy, pravděpodobně si ze cvičení kromě pocitu neúspěchu příliš neodnesete a hrozí vám vyloučení ze cvičení.

Věnujte prosím pozornost jak fungování fakultních systémů (počítačů v učebnách, studentského serveru Aisa), tak nastavení svých strojů. Mít vše rozjeté na svých strojích se vám bude hodit při řešení domácích úkolů, pokud byste například skončili doma nemocní. Být schopný programovat ve vlaku či jinde bez internetu je též velmi užitečné.

Nulté cvičení je velmi nestandardní. Probíhá pro všechny studenty v prvním týdnu (nejde tedy o 14denní blok) a slouží především k nastavení programů nutných pro fungování v tomto předmětu. Cvičení trvá necelou hodinu a nezabývá se náplní předmětu, ale má za úkol vás seznámit s počítačovou učebnou B130 a jejím programovým vybavením, aby se na prvním cvičení nemusely řešit problémy technického rázu. Čas na něm můžete také využít ke konzultaci problémů s instalací na svých strojích – později na to již na cvičeních čas nebude, takže se budete muset spokojit s konzultacemi na diskusním fóru.

Vysvětlivky I nultá kapitola sbírky je trochu zvláštní – slouží totiž zároveň jako seznámení se sbírkou samotnou. Jednou z věcí zasluhujících vysvětlení je význam piktogramů objevujících se u některých příkladů:

»= Symbolem »= jsou označeny ty příklady, které by se měly stihnout na cvičení. Příklady bez tohoto symbolu tak slouží spíše pro domácí studium a přípravu. Mimochodem, přesný význam tohoto symbolu se dozvíte případně na navazujícím předmětu IB016 Seminář z funkcionálního programování. Pokud mu chcete nějak říkat, můžete používat termín *bind*. Ještě více mimochodem, podobnost s logem Haskellu není náhodná.

Tužkou (✎) jsou označeny příklady, které byste měli být schopni vyřešit správně s použitím pouze tužky a papíru, stejně jako se to po vás bude chtít u zkoušky.[1] Interpret doporučujeme použít až ke kontrole.

[1] Nepodlehněte však iluzi, že před zkouškou stačí projít tužkou označené příklady a zelenou známku máte v kapse. Piktogram totiž nijak nesouvisí s typem příkladů objevujících se u zkoušky.

Démantem () označujeme příklady, které sice nejsou v plánu cvičení, ale přesto je velmi doporučujeme vyřešit, neboť je považujeme za zajímavé či zvláště přínosné. Očekáváme, že tyto příklady budete řešit pravidelně v průběhu semestru a mohou vám pomoci při řešení odpovídajících domácích úkolů.

Mírně obtížnější, ale o to poutavější příklady pak nesou hvězdičku () či několik hvězdiček podle obtížnosti či pracnosti. Trojhvězdičkové příklady mohou sahat i za rámec tohoto předmětu.

K vybraným příkladům existují videa demonstrující postup řešení příkladu. Tato videa jsou alternativou k živému vysvětlení od cvičícího a také slouží k jeho připomenutí.

Etudy

Každá kapitola na začátku obsahuje několik základních příkladů, tzv. etud. Očekává se, že tyto příklady si vyřešíte ještě *před* samotným cvičením. Slouží především k ozkoušení syntaxe jazyka Haskell, případně připomenutí jeho standardních funkcí – tedy přibližně to, co po vás chce fialový rámeček na začátku.

V případě problémů či dotazů se zeptejte na diskusním fóru.

> *Zelené rámečky obsahují rady, doporučení a tipy na další zdroje ke studiu.*
>
> **Pan Fešák doporučuje:** K většině příkladů se na konci sbírky (případně kapitoly, čtete-li tyto samostatně) nachází řešení. Podporuje-li váš prohlížeč dokumentů odkazy, můžete klepnutím na číslo příkladu na jeho řešení skočit.

Etuda 0.η.0 Zkuste si to!

Etuda 0.η.1 Seznamte se s nultou kapitolou sbírky a s organizačními pokyny k předmětu v interaktivní osnově.

Etuda 0.η.2 **Nainstalujte si GHC na svůj počítač.** Standardem pro tento semestr je překladač Glasgow Haskell Compiler (GHC) verze 8.4 nebo vyšší. Doporučujeme instalovat nejnovější dostupný pro váš systém, ideálně tedy verzi 9.2.

Linux Pokud je GHC verze nejméně 8.4 k dispozici v repozitářích vaší distribuce, instalujte přímo z repozitáře. Níže uvádíme návody pro některé běžné distribuce, ve všech případech ukazujeme jak nainstalovat GHC z příkazové řádky pod uživatelem `root`.

- **Ubuntu/Debian** Pokud máte dostatečně novou verzi operačního systému (Debian alespoň 10 = buster = oldstable, Ubuntu alespoň 20.04), stačí vám nainstalovat balík `ghc`:

  ```
  apt install ghc
  ```

  Pro starší verse, či pokud chcete novější versi GHC najdete návod v řešení tohoto příkladu.

- **Fedora**: Ve všech podporovaných verzích Fedory je již dostatečně nové GHC, stačí nainstalovat balík `ghc`:

  ```
  dnf install ghc
  ```

Windows Na Windows je možné instalovat GHC ze stránek GHC, nebo alternativně do Windows Subsystem for Linux (WSL; konkrétně ideálně verzi Ubuntu 20.04 LTS).

Zde popíšeme pouze první možnost, pro tu druhou je třeba aktivovat WSL a následně postupovat podle návodu pro Ubuntu výše.

- Stáhněte GHC ze stránek GHC („Windows 64-bit (x86_64) (GMP)“).
  - rozbalte stažený archiv (například pomocí 7-Zip)
  - přesuňte složku `ghc-X.Y.Z` z rozbaleného archivu do `C:\` (`X.Y.Z` zde odpovídá verzi staženého GHC, např. `ghc-9.2.3` – chcete-li si práci usnadnit, můžete si složku přejmenovat na `ghc` a příponu `-X.Y.Z` v rámci návodu směle ignorovat)
- Dále potřebujeme nastavit proměnné prostředí, abychom mohli GHC/GHCi spouštět kdekoliv v rámci systému.
  - Jděte do „Upravit proměnné prostředí systému“ (najdete přes vyhledávání nebo „Počítač“ → „vlastnosti“ → „upřesnit nastavení systému“ → „upravit proměnné prostředí“);
  - klikněte na „proměnné prostředí“
  - v části „systémové proměnné“ vyberte proměnnou **path** a zvolte upravit
  - klikněte na „nový“ a přidejte cestu `C:\ghc-X.Y.Z\bin` (nebo jinou cestu kam jste GHC přesunuli)
  - uložte

Pro ověření spusťte příkazový řádek (stiskněte win + R a zadejte příkaz `cmd`) a do příkazového řádku napište `ghci`. Mělo by se spustit GHCi ve verzi odpovídající staženému archivu.

macOS Na macOS jsou dvě možnosti instalace – přes Homebrew (doporučené, vyžaduje nainstalovat Homebrew, které se vám však bude hodit pro instalaci mnoha dalších nástrojů při studiu na FI) a nebo pomocí nástroje `ghcup`. V obou případech instalace probíhá z příkazové řádky.

Homebrew instalace

- otevřete Terminal pod `/Applications/Utilities/`
- nainstalujte si Homebrew z příkazové řádky příkazem z `https://brew.sh/`
- nainstalujte GHC příkazem `brew install ghc` (opět z příkazové řádky)
- spusťte `ghci` – macOS jej prvním spuštění pravděpodobně zablokuje, povolte jej tedy v nastavení (viz též apple support)
- dále by již mělo být možné GHCi spouštět normálně

Instalace pomocí ghcup GHC se v tomto případě instaluje pouze pro toho uživatele, který provede níže zmíněný postup.

- jděte na `https://www.haskell.org/ghcup/` a zkopírujte příkaz uvedený na této stránce
- otevřete Terminal pod `/Applications/Utilities/` a vložte do něj příkaz z webu `ghcup`
- postupujte podle instrukcí instalátoru, nezapomeňte si nechat automaticky upravit `PATH` (to zajistí, že GHC budete moci spustit bez zadání celé cesty k binárce v `/.ghcup`)
- restartujte terminál, pak by již mělo být možné GHCi spustit

Etuda 0.η.3 **Nainstalujte si vhodný editor čistého textu na svůj počítač.** Pro psaní kódu v jazyce Haskell, který v tomto předmětu používáme, budete potřebovat editor schopný pracovat se zdrojovým kódem. Jde o editor, který ukládá čistý, neformátovaný text a případně

má funkce, která pomáhají programátorům v pochopení kódu – především zvýrazňování syntaxe (tedy rozlišení různých konstruktů v kódu barvami).

> **Jindřiška varuje:** Word ani LibreOffice Writer **nejsou** editory čistého textu.

Na různých systémech jsou obvyklé různé editory zdrojového kódu, zde uvedeme 2 příklady, které mohou fungovat na všech platformách, můžete však používat i jiné. Ve všech editorech doporučujeme pro Haskell nastavit odsazování pomocí mezer (nikoli tabulátorů), což je v Haskellu standardní konvence. Navíc kombinování odsazování mezerami a tabulátory vede v Haskellu ke špatně pochopitelným chybám. Pokud již máte svůj oblíbený editor zdrojových kódů (který umí pracovat s Haskellem), můžete zbytek této sekce ignorovat.

Visual Studio Code Visual Studio Code (také VS Code, nebo jen code) je zdarma dostupný multiplatformní editor zdrojového kódu od firmy Microsoft. Informace k jeho instalaci na vaši platformu naleznete na webu výrobce, na některých Linuxových distribucích je VS Code k dispozici rovněž v repozitářích (např. jako `code` na Arch Linuxu).

Základní nastavení

- Pod „View“ → „Extensions“ si vyhledejte Haskell a nainstalujte „Haskell Syntax Highlighting“
- Následně již bude fungovat zvýrazňování syntaxe pro soubory s příponou .hs (v pravém dolním rohu byste měli vidět „Haskell“)
- V pravé dolní části si zkontrolujte, že máte nastaveno odsazování mezerami: „Spaces: N“ (nikoli „Tab Size: N“), kde N označuje délku odsazení (pro Haskell je vhodné 4).
- VS Code má i integrovaný terminál, který vám může zjednodušit práci tím, že budete mít editor i terminál v jednom okně.
  - Terminál aktivujete ve „View“ → „Terminal“
  - Nezapomeňte se před spuštěním ghci přepnout do adresáře, v němž máte uložen editovaný soubor (pomocí cd CESTA).
  - Nezapomínejte, že pro to, aby se změny v kódu projevili, je třeba soubor uložit a v ghci provést reload (:r).
- **Pozor, neinstalujte si žádná „pokročilejší“ Haskell rozšíření (například to jménem „Haskell“)** – tato rozšíření vyžadují komplexnější nastavení a často předpokládají, že budete tvořit tzv. balíčky. Pro naše použití tak prakticky způsobují jen komplikace a chyby editoru.

Tip: Pokud si ve VS Code otevřete pracovní složku, kde chcete programovat v Haskellu pomocí „File“ → „Open Folder“, integrovaný terminál se vám pak také otevře v této složce.

Gedit Gedit je výchozí textový editor v několika variantách Linuxové distribuce Ubuntu a na všech Linuxových distribucích by měl být dostupný (obvykle v balíku `gedit`) a je rovněž dostupný pro Windows a macOS. Gedit doporučujeme používat především pokud chcete co možná nejjednodušší editor.

Základní nastavení

- Zvýrazňování syntaxe funguje v Geditu automaticky pro soubor s příponou .hs, není tedy třeba nic nastavovat.
- Doporučujeme zapnout si zobrazování čísel řádků (přes druhou položku zprava na spodní liště).
- Doporučujeme zapnout odsazování mezerami (v Haskellu se nedoporučuje používat tabulátory a už vůbec ne je kombinovat s mezerami). Na spodní liště vpravo klikněte na „Tab Width: N“ a zaškrtněte „Use Spaces“. Může být rovněž dobrý nápad snížit výchozí odsazení na 4 mezery.

Etuda 0.η.4 intro Zhlédněte záznam úvodního democvičení. Můžete k tomu použít odkaz pod číslem tohoto příkladu; vede do studijních materiálů předmětu v ISu.

*
* *

Po etudách následují příklady určené k řešení na cvičení nebo samostatně po cvičení, například jako příprava na domácí úkol. Nic vám ale samozřejmě nebrání se s příklady seznámit ještě před cvičením.

0.1 Základní potřebné programy a nastavení

> Je v podstatě jedno, zda budete na cvičeních používat svůj počítač nebo školní. Je však silně doporučené abyste měli vše potřebné nainstalované u sebe (kvůli přípravám a domácím úkolům), ale zároveň je i vhodné, abyste uměli používat školní počítače.

Př. 0.1.1 »= **Základní využívání školních počítačů.** Přihlaste se na školní počítač (pomocí svého xloginu a fakultního hesla). Po přihlášení spusťte terminál (buď jej vyhledejte, nebo pomocí klávesové zkratky Ctrl+Alt+T). Po spuštění by měl terminál pracovat ve vašem domovském adresáři (`/home/xLOGIN`). Příkazem `pwd` si vypište aktuální adresář.

> **Základní příkazy příkazové řádky**
> Tyto příkazy by měly fungovat v podstatě na všech platformách, včetně Windows (jejich přesný výstup v případě, že něco vypisují, se však může lišit). Na Windows však může být nutné použít PowerShell místo příkazové řádky (neboli `cmd`).
>
> `ls` vypíše obsah aktuálního adresáře. Případně `ls -A` vypíše adresář včetně souborů a složek, jejichž jméno začíná tečkou.
>
> `mkdir` slouží k vytvoření adresáře. Argumentem je cesta k adresáři, který chcete vytvořit: `mkdir ib015`. Pokud je potřeba vyrobit mezilehlé adresáře, lze použít `mkdir -p longer/path/to/ib015`.
>
> `cd` přepíná adresáře. Jediný argument je cesta k adresáři, kam přepnout. Např. `cd ib015` pro přepnutí do adresáře `ib015`, `cd ..` pro přechod nahoru v adresářové struktuře nebo `cd ~` (vlnovka) pro skok do domovského adresáře (mimo Windows i jen `cd`).

`ssh` slouží pro vzdálený terminálový přístup na počítač (typicky se systémem Linux/Unix). Argumentem je adresa počítače a případně login: `ssh xLOGIN@aisa.fi.muni.cz` (s vaším loginem). Při prvním přihlášení je třeba potvrdit klíč stroje. Heslo se při zadávání nezobrazuje. Pokud je váš lokální login stejný, jako ten, do kterého se chcete přihlásit, nemusíte jej uvádět: `ssh aisa.fi.muni.cz` (také po vyřešení 0.3.3 níže).

`scp` slouží ke kopírování souborů mezi počítači. První argument je vždy zdroj, druhý cíl. Jedna z cest je lokální, jedna může být ve tvaru `login@stroj:cesta` a pak představuje cestu na daném stroji. Cílem může být i složka; cesta by pak měla končit lomítkem a složka musí na daném stroji existovat. Např. `scp Foo.hs xLOGIN@aisa.fi.muni.cz:` pro kopírování souboru `Foo.hs` do domovského adresáře na Aise (relativní cesty jsou vzhledem k domovskému adresáři). Aktuální adresář je označen „`.`“ (tečka), má-li být cílem ten.

`exit` ukončení příkazové řádky nebo sezení SSH. Můžete použít i Ctrl+D.

`cp` jako `scp`, ale jen lokálně.

`mv` přesun/přejmenování souboru, argumenty jako u `cp`.

`rm/rmdir` maže soubory, respektive prázdné složky zadané jako argumenty.

`nano` jednoduchý editor textu v terminálu (z Aisy nemůžete spustit grafický editor). Příkazy pro ovládání se zobrazují dole, ^ znamená Ctrl. Na vašem počítači možná nebude.

Př. 0.1.2 Vytvořte si ve svém domovském adresáři adresář `ib015`. Můžete využít terminál nebo grafický správce souborů.

Př. 0.1.3 Většinu úkolů budete vypracovávat v textovém editoru podle svého výběru. Proto si jej nastavte podle instrukcí v etudě 0.η.3, jen s tím rozdílem, že oba editory jsou již nainstalované a stačí je nastavit.

Př. 0.1.4 V adresáři `ib015` vytvořete nový soubor `Sem0.hs` s následujícím obsahem:

```
hello = "Hello, world"
```

0.2 Používání GHCi

Pokud si kapitolu procházíte samostatně, tak v této a následující sekci již přijde vhod democvičení (`intro`), je proto dobrý nápad jej nyní zhlédnout. Může být přínosné si nahrávku zastavovat a postupovat podle ní, případně se k nejasným částem vracet.

Př. 0.2.1 V terminálu v adresáři `ib015` si příkazem `ghci` spusťte interpret Haskellu.

Komu zrovna zadáváte text, poznáte podle tzv. výzvy, neboli promptu. Prompt GHCi má obvykle tvar `Prelude>`, `Main*>` nebo `ghci>` – sem zadáváte výrazy jazyka Haskell a příkazy interpretu (ty, co začínají dvojtečkou). Prompt terminálu nabývá rozličných podob, ale obvykle končí znakem dolaru (např. `login@počítač:~/ib015 $`). Sem zadáváte např. příkazy pro práci se soubory a pohyb po adresářích.

Př. 0.2.2 V interpretu si zkuste vyhodnotit jednoduché aritmetické výrazy, tedy využít ho jako kalkulačku.

Př. 0.2.3 Do GHCi načtěte soubor `Sem0.hs` a nechte si vypsat konstantu `hello`. Následně si od interpretu vyžádejte její typ.

Př. 0.2.4 Do souboru `Sem0.hs` přidejte konstantu `myNumber` typu `Integer` a nastavte ji na svou oblíbenou hodnotu. Po uložení souboru jej znovu načtěte do GHCi a konstantu vypište. Interpret následně ukončete.

> **Pan Fešák doporučuje:** Pokud se někdy program nechová tak, jak očekáváte, ačkoli jste si jisti, že tentokrát už jste ho určitě opravili, zkontrolujte, že jste soubor uložili a znovu načetli do interpretu. Je to jedna z nejčastějších chyb.

0.3 Vzdálené připojení na fakultní počítače

> Používání vzdáleného připojení přes SSH není v tomto předmětu nezbytně nutné, ale může se vám hodit a určitě jej budete potřebovat jinde. Pan Fešák proto doporučuje se ho naučit již nyní. Vzdálené připojení vám umožní používat GHCi a další nástroje na počítačích na FI a dostat se k vašim kódům ze cvičení odkudkoli.

Na školní stroje se systémem Linux se lze připojit pomocí SSH. Na Linuxu či macOS se obvykle jako klient SSH používá příkaz `ssh`. Pokud klient nemáte, mělo by být možné jej nainstalovat z balíčků (bude se nejspíš jmenovat `openssh-client` nebo `openssh`). Na Windows 10 je příkaz `ssh` k dispozici v PowerShellu. Záložní možností je pak klient Putty.

K přihlašování z vnějšku fakulty slouží server `aisa.fi.muni.cz` (ze sítě FI dostupný i jako `aisa`). Přes tento počítač se pak případně dá dostat i k jiným, ale to nebývá potřeba, protože váš domovský adresář je sdílen přes všechny fakultní linuxové stroje.

> **Co se kde děje?** Při práci se vzdáleným počítačem může být trochu nepořádek v tom, co se děje u vás a co na vzdáleném stroji. Měli byste to být schopni poznat podle promptu, tedy textu, který příkazová řádka vypisuje před místem, kam píšete příkaz. Na Aise byste měli (pokud jste si to nezměnili) vidět něco jako `aisa:/home/xLOGIN>$`. V promptu je tedy vidět i cesta k adresáři, v němž se aktuálně nacházíte.
>
> Aby v tom nebyl nepořádek v našich příkladech, používáme konvenci, že příkazy spouštěné na Aise začínají `(aisa)$`, zatímco ty spouštěné u vás, začínají `(local)$` (pokud je to jedno, bude tam jen `$`). Ani jeden z těchto řetězců však nepatří na příkazovou řádku – slouží jen k označení příkazů, které můžete spustit.

> Nápovědu k příkazům a jejich volbám si vyžádejte příkazem `man` nebo volbou `--help`:
>
> ```
> $ man rm
> $ cd --help
> ```

Př. 0.3.1 Přihlaste se ze svého nebo školního počítače k serveru `aisa.fi.muni.cz`.

Př. 0.3.2 Pomocí příkazu `scp` si na svůj počítač zkopírujte z Aisy zdrojový kód ze cvičení. U sebe jej upravte a pošlete zpět na Aisu. Pokud jste pracovali na svém počítači, udělejte to obráceně – kód zkopírujte na Aisu, tam ho upravte (třeba pomocí `nano`) a zkopírujte zpět.

Alternativně můžete vyzkoušet některý z klikacích nástrojů: na Windows WinSCP, v Linuxových grafických správcích souborů hledejte volbu „Připojit k serveru“ nebo podobnou.

Př. 0.3.3 Nastavte si klient SSH tak, aby pro připojení k Aise stačilo zadat `ssh aisa`. Stejnou zkratku je pak možné využívat při kopírování souborů mezi počítači: `scp Sem0.hs aisa:ib015/`.

> **Pan Fešák doporučuje:** Než navštívíte svůj oblíbený vyhledávač, vzpomeňte si na příkaz `man`. Zde se bude hodit `man ssh_config`.

Př. 0.3.4

> Tento krok je určen pokročilejším uživatelům a sahá daleko nad rámec předmětu.

Pokud nedisponujete dvojicí klíčů SSH pro asymetrické šifrování, vytvořte si ji. Přidejte na Aise svůj veřejný klíč mezi autorizované, abyste při přihlašování ze svého počítače nemuseli psát heslo.

0.4 Práce s dokumentací

Tak jako u ostatních programovacích jazyků je většina funkcí a typů zdokumentovaná. Primárním zdrojem dokumentace pro jazyk Haskell v rozsahu našeho kurzu je webová dokumentace základního modulu `Prelude`. Dále můžete využít lokální mirror vyhledávače Hoogle, kde můžete vyhledávat funkce podle názvu nebo typu.[3]

> **Jindřiška varuje:** Naučit se pracovat s dokumentací je nevyhnutelné pro libovolný programovací jazyk, nejenom pro Haskell. Čím dříve se naučíte číst dokumentaci, tím budete mít lehčí život nejen tu, ale i v dalších předmětech.

Př. 0.4.1 Pomocí vyhledávače funkcí v jazyce Haskell najděte všechny funkce, které mají typ `Bool -> Bool -> Bool` a jsou v balíčku `base`.

*
* *

> **Jindřiška varuje:** Ať už budete používat vlastní či školní počítač, je naprosto nezbytné, abyste před prvním cvičením zvládli jeho obsluhu alespoň v rozsahu fialového rámečku na začátku následující kapitoly.

> *Fialové rámečky na konci cvičení shrnují probrané koncepty. Můžete si s jejich využitím ověřit, jestli všemu ze cvičení rozumíte, nebo je potřeba se k některému tématu vrátit.*
>
> **Na konci cvičení byste měli zvládnout:**

[2] Na Aise je nainstalované příliš staré GHC, které se od toho námi používaného liší typy některých seznamových funkcí.

[3] Obě tyto služby pro vás hostujeme lokálně, protože ty oficiálně se ukázaly být pro naše účely neoptimální – originální dokumentaci najdete na Hackage a vyhledávání i v nestandardních balíčcích na hoogle.haskell.org.

- vytvořit textový soubor s Haskellovým kódem;
- spustit v terminálu GHCi a načíst do něj vytvořený soubor;
- pracovat v terminálu vzdáleně na školním počítači Aisa;
- pracovat s webovou dokumentací.

Řešení

Řeš. 0.η.0 Gratulki.

Nenechávejte si ale řešení prozradit příliš brzy. Má-li mít práce se sbírkou smysl, je potřeba se nad každým příkladem nejprve zamyslet a následně ho samostatně vypracovat. Dokonce ani když se vám implementace nedaří, nenakukujte hned do řešení – nejvíce se naučíte, pokud se nad příkladem sami trochu potrápíte. Samozřejmě klidně konzultujte přednášku nebo již hotové příklady.

Teprve až příklad zdárně vyřešíte, můžete nahlédnout do vzorového řešení. Občas se můžete dozvědět jiný (a v něčem třeba elegantnější) postup nebo nějaké tipy a triky.

Klepnutím na číslo řešení přeskočíte zpátky na příklad.

Řeš. 0.η.2 **Návod pro starší Debian a Ubuntu**

1. podle distribuce:
   (a) Pokud máte Debian, postupujte podle návodu na `https://downloads.haskell.org/debian/` („Debian Instructions“).
   (b) Pro Ubuntu si přidejte PPA repozitář `https://launchpad.net/~hvr/+archive/ubuntu/ghc` – čtěte sekci „Adding this PPA to your system“).
2. Nainstalujte si `ghc-9.0.1` (nebo novější):

   ```
   apt install ghc-9.0.1
   ```

3. Výsledné GHCi/GHC pak naleznete v cestě `/opt/ghc/bin/ghci`, resp. `/opt/ghc/bin/ghc`.
4. Tuto cestu je vhodné přidat do proměnné prostředí `PATH` abyste mohli interpret spouštět jen jako `ghci`: otevřte v libovolném textovém editoru (třeba VS code, `gedit`, či `nano`) soubor `.bashrc` ve svém domovském adresáři (nenechte se zmást tím, že není vidět ve výpisu `ls`, soubory začínající tečkou `ls` samo o sobě nevypisuje) a na jeho konec následující řádek:

   ```
   export PATH="$PATH:/opt/ghc/bin"
   ```

   Nyní již můžete interpret spouštět přímo, bez zadání celé cesty.

Řeš. 0.1.2 Vytvořit jej můžete buď v terminálu (příkazem `mkdir` *`jméno`* a následně do něj přepnout pomocí `cd` *`jméno`*), nebo v grafickém správci souborů, který by se měl otevřít v domovském adresáři a nový adresář by mělo být lze vytvořit pravým myšítkem nebo přes panel nabídek.

Řeš. 0.1.4 Spusťte textový editor (například Gedit nebo VS Code – `code` na fakultních PC) a napište či překopírujte do něj text. Nezapomeňte soubor uložit do správného adresáře.

Alternativně můžete soubor vytvořit i v terminálu, například editorem `nano`, který vám v dolní části zobrazuje klávesové zkratky, kterými se ovládá (^ znamená Ctrl).

Jiným oblíbeným terminálovým editorem je `vim`. Jeho obsluha vypadá zpočátku krkolomně, ale dá se rychle naučit: zkuste příkaz `vimtutor`, nebo `vimtutor cs`, případně `vimtutor sk`. (Ukončit je lze zadáním ‘:q‘ a odentrováním.)

Řeš. 0.2.2 Zadejte výraz a stiskněte Enter. Např. `40 + 2` se vyhodnotí na `42`. Vyhodnocení výrazů zapisujeme ve studijních materiálech pomocí lomené šipky s hvězdičkou: `40 + 2` $\rightsquigarrow^*$ `42`.

Další příklady:

- `4 * 10 + 2` ⇝* `42`
- `2 ^ 8` ⇝* `256`
- `(3 + 4) * 6` ⇝* `42`

Řeš. 0.2.3 Načíst soubor lze buď pomocí `:load Sem0.hs` nebo jen `:l Sem0.hs` (pokud se nachází ve stejném adresáři, kde jsme spustili GHCi), nebo tak, že spustíme GHCi přímo s tímto názvem souboru (`ghci Sem0.hs`). Konstantu vypíšeme prostě tak, že ji napíšeme do GHCi a stiskneme enter: `hello` ⇝* `"Hello, world"`. Její typ pak zjistíme příkazem interpretu `:t`, který jako parametr bere libovolný výraz: `:t hello` ⇝* `hello :: [Char]`.

Řeš. 0.2.4
```
myNumber :: Integer
myNumber = 42
```

Znovunačtení lze provést příkazem `:r`, ukončení `:q`.

V případě, že vytvoříte proměnnou jenom v interpretu, pak ta přestane po ukončení interpretu existovat.

Řeš. 0.3.1 Na Linuxu, macOS a Windows s dostatečně novým PowerShellem by mělo stačit zadat do příkazové řádky (či PowerShellu) `ssh xLOGIN@aisa.fi.muni.cz` (samozřejmě s vaším loginem) a následně zadat své fakultní heslo. Nenechte se překvapit tím, že při zadávání hesla nevidíte žádné hvězdičky; v terminálu se hesla běžně zadávají „naslepo“. Ocitnete na Aise a můžete tam používat příkazovou řádku jako na cvičení. Na Windows s Putty musíte nastavit sezení (*session*), kde jako hostitele uvedete `xLOGIN@aisa.fi.muni.cz`.

Pro trvalé přidání modulu můžete příkaz vložit do svého souboru `.bashrc`:

```
(aisa)$ echo "module add ghc" >> ~/.bashrc  # Nebo textovým editorem
```

Verze GHCi se vypíše po spuštění interpretu, nebo po zadání příkazu `ghci --version`.

Řeš. 0.3.2 Aby první následující příkaz fungoval, je potřeba být lokálně v adresáři, kde je soubor `Sem0.hs` a na Aise musí existovat adresář `ib015`.

```
(local)$ scp Sem0.hs xLOGIN@aisa.fi.muni.cz:ib015/
(aisa)$ nano Sem0.hs        # Nebo jiný způsob editace
(local)$ scp xLOGIN@aisa.fi.muni.cz:ib015/Sem0.hs Sem0_edited.hs
```

Řeš. 0.3.3 Do souboru `~/.ssh/config` na svém počítači vložte (a nastavte svůj login):

```
Host aisa
    HostName %h.fi.muni.cz
    User xLOGIN
```

Řetězec `%h` v nastavení nechte, ten představuje specielní sekvenci, za kterou se doplní jméno napsané za `Host` – takových jmen může být více (oddělených mezerou). Pokud daný soubor neexistuje, vytvořte jej (i na Windows může být jednodušší vytvořit minimálně složku `~/.ssh` pomocí příkazové řádky, ta by však existovat měla). Soubor `config` nesmí mít žádnou příponu. Část `~` v cestě značí domovský adresář. Složka `.ssh` nebude ve výpisu přes `ls` vidět, pokud nepoužijete `ls -A`, což způsobí zobrazení skrytých souborů – těch co začínají `.` (tečkou).

Řeš. 0.4.1 Jděte na `https://hoogle.haskell.org/`. V rozbalovací nabídce u vyhledávacího políčka zvolte `package:base` a do vyhledávacího políčka zadejte hledaný typ. Po chvíli by vám vyhledávač měl najít funkce `(&&)`, `(||)`, `(==)` a `(/=)`. Poslední dvě mají obecnější typ, ale fungují i pro argumenty typu `Bool` – Hoogle vyhledává i obecnější funkce, do kterých lze za typové proměnné dosadit požadované typy.[4] U každé funkce také vidíte, ve kterém

balíčku (`base`) a modulu (`Prelude`[5]) se nachází, a také dokumentační text, pokud je uveden. Kliknutím na funkci se dostanete do její dokumentace v prvním modulu, kde byla nalezena.

[4] V tomto případě je navíc typ omezený typovým kontextem `Eq =>`, což znamená, že za `a` lze dosadit pouze porovnatelné typy, což `Bool` splňuje.

[5] `Prelude` je základní Haskellový modul, který je vždy k dispozici.